SYSTEM, METHOD, AND ARTICLE OF MANUFACTURE FOR EMBEDDED KEYWORDS IN VIDEO

FIELD OF THE INVENTION

The present invention relates to video and multimedia and more particularly to embedded keywords in video.

BACKGROUND OF THE INVENTION

Multiple sources of information are available for recipients at their homes and offices, including traditional broadcast information sources such as standard broadcast radio and television (available by transmission to a receiver in the home or office), and cable and satellite television (possibly encrypted, scrambled, or otherwise restricted for delivery); traditional switched information sources such as telephone services; and traditional recorded information sources such as audio and video recorded tapes, compact discs and laser discs, and other recorded media.

In addition to these traditional information sources, more recently available information sources include enhanced television services such as "closed caption" (sometimes made available using a television vertical blanking interval or other broadcast or narrowcast data delivery paths), "picture-in-picture" and related techniques for presenting more than one television channel for simultaneous viewing, presentation of textual information using a cable or satellite television system, and videoconferencing; enhanced telephone services such as paging, voicemail or voice messaging, voice response systems, and cellular telephone service; and networking services such as electronic mail, access to the world wide web, and other internet and intranet resources.

Some of these enhanced services have included transmitting information in a variety

of formats, including stock quotations and similar financial data in real time, textual information serving to complement a television broadcast (such as subtitles), and information in computer formats (such as HTML code). However, even in circumstances where multiple information sources share a common communication channel (such as the television vertical blanking interval or a radio or television subcarrier), in general each information source has been associated with its own information format, its own selection of which information is to be presented, and its own technique for retrieving and using that information.

Additionally, as television viewers become more familiar with on-line services, such as the internet and the World Wide Web, they are demanding access to on-line information and content related to the television and content and television-advertised companies and products. This information is becoming more readily available to the internet user. An increasing number of television broadcasters, advertisers, and organizations have created their own internet locations (i.e., websites), and many have begun to display their web site address, (i.e., their Universal Resource Locator (URL) on TV ads and programs (broadcast events) to encourage viewers to access their website for more information. However, viewers typically cannot easily recall or identify products, services, or other content shown during a particular video performance or broadcast event because products or services are typically displayed for only a brief time. A tool is needed to allow the viewer to easily access websites, and to immediately navigate to the desired site, i.e., switch to the displayed website as if it were a television channel, or to "bookmark" the performances or broadcast event, i.e., to mark one or more performances or broadcast events so the viewer can later recall these events and access websites containing information relating with these products, services, or other content they viewed during the performance or broadcast.

SUMMARY OF THE INVENTION

A system, method and article of manufacture are provided for embedding keywords in video (digital video, in particular). Multimedia content is displayed comprising one or more digital video images. One or more keywords are associated with at least one of the video images. A network is utilized to retrieve information relating to the keywords, by use of a search engine, for example, that executes a keyword search for one or more sites in the network that relate to the keyword. The retrieved information relating to the keyword is then displayed. This retrieved information may be displayed in a summary form that summarizes the sites found during the search relating to the keyword.

In an embodiment of the present invention, the keywords associated with the video images of the multimedia content may also be displayed. As an option, the keywords may be displayed adjacent to their associated video images. In another embodiment of the present invention, one of the displayed video images of the content may be selected to thereby display the keywords associated with the selected video image. In such an embodiment, this may be accomplished by displaying a pointer and then permitting a user to position the pointer over the displayed image to select the particular video image to thereby display the associated keywords. As a further option, the user may also be able to select one of the displayed keywords with the pointer to execute a search for information relating to the selected keyword utilizing the network.

In a further embodiment of the present invention, the retrieved information may also be indexed based on the associated keywords. In yet another embodiment of the present invention, a user profile of a user may be obtained so that the retrieved information may include information related to the user profile of the user. In an aspect of the present invention, the video images of the multimedia content may be carried/transmitted in a first channel and the keywords may be carried/transmitted in a second channel. In yet another aspect of the present invention, content may utilize the Advanced Television Enhancement Forum (ATVEF) Content Specifications.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure **1** is a schematic diagram of one possible hardware implementation by which the present invention may be carried out in accordance with an embodiment of the present invention;

Figure **2** illustrates a flowchart for a process for embedding keywords in video in accordance with an embodiment of the present invention;

Figure **3** is a simplified block diagram of a search engine system that operates in accordance with an embodiment of the invention;

Figure **4** presents an overview of the functionality provided by the search engine system in accordance with an embodiment of the present invention;

Figure **5** illustrates operations performed to create a filter used to search broadcast information and Internet information in accordance with an embodiment of the present invention;

Figure **6** illustrates the operations performed to activate a search in accordance with an embodiment of the present invention;

Figure **7** illustrates the operations performed to render on a display the query of the results of the search performed in accordance with an embodiment of the present invention;

Figure **8** is a table of search results for a two-word query using a relevance-ranking algorithm in accordance with an embodiment of the present invention;

Figure **9** is a flowchart showing a relevance ranking method in accordance with an embodiment of the present invention;

Figure **10** is flowchart showing a method that may be used to produce an adjusted relevancy ranking score in accordance with an embodiment of the present invention;

Figure **11** illustrates an illustrative event marking system for marking viewer selected performances and broadcast events that may be used to bookmark portions of a program or broadcast for allowing the user to return to locations to retrieve keywords from that point in the program or broadcast in accordance with an embodiment of the present invention;

Figure **12** illustrates an illustrative automated custom program scheduling and display method using the event marking system of Figure **11** in accordance with an embodiment of the present invention;

Figure **13** is high level diagram of a system for computerized searching of data in accordance with an embodiment of the present invention;

Figure **14** is a high level diagram of a method for searching, indexing, and displaying data stored in a network in accordance with an embodiment of the present invention; and

Figure **15** is a high level diagram of a method for searching, indexing, and displaying data stored in a network using a cluster generation algorithm in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION

In older video broadcast systems, video is sent occupying the full bandwidth of the transmission channel. In newer digital technology a secondary channel can be sent along with the video to provide additional functionality. The present invention allows for the secondary channel to provide advanced features with extremely low bandwidth. The present invention provides keywords that accompany the video that may be later searched upon to get more information about the video content. In addition, a meta language may be utilized to add commands on how the data is to be processed. The meta language may be a pseudo-HTML or some other meta language. This can be used as a form of advertisement or simply a starting place to provide more information to the viewer. Given a keyword, this may also be used to submit a query to a search engine and obtain even more information. For example if a viewer was watching the travel channel and the special was on Hawaii, then, the keyword "Hawaii" may be presented to the user so that information may be retrieved and displayed to the user regarding the destination, travel packages etc based on the search results. In addition, if the airline trip is being shown to the viewer then an airline "American Airlines" may be use as a keyword which is shown to the user as well.

Figure **1** is a schematic diagram of one possible hardware implementation by which the present invention may be carried out. As shown, the present invention may be practiced in the context of a personal computer such as an IBM compatible personal computer, Apple Macintosh computer or UNIX based workstation. Additionally, the present invention may be carried out with what is known as a set-top box, Web TV-type device and the like.

A representative hardware environment is depicted in Figure **1**, which illustrates a typical hardware configuration of a workstation in accordance with one embodiment having a central processing unit **110**, such as a microprocessor, and a number of other units interconnected via a system bus **112**. The workstation shown in Figure **1**

includes a Random Access Memory (RAM) **114**, Read Only Memory (ROM) **116**, an I/O adapter **118** for connecting peripheral devices such as disk storage units **120** to the bus **112**, a user interface adapter **122** for connecting a keyboard **124**, a mouse **126**, a speaker **128**, a microphone **132**, and/or other user interface devices such as a touch screen (not shown) to the bus **112**, communication adapter **134** for connecting the workstation to a communication network **135** (e.g., a data processing network) and a display adapter **136** for connecting the bus **112** to a display device **138**.

The workstation typically has resident thereon an operating system such as the Microsoft Windows NT or Windows/95 Operating System (OS), the IBM OS/2 operating system, the MAC OS, or UNIX operating system. Those skilled in the art will appreciate that the present invention may also be implemented on other platforms and operating systems.

A preferred embodiment of the present invention is written using JAVA, C, and the C++ language and utilizes object oriented programming methodology. Object oriented programming (OOP) has become increasingly used to develop complex applications. As OOP moves toward the mainstream of software design and development, various software solutions require adaptation to make use of the benefits of OOP.

OOP is a process of developing computer software using objects, including the steps of analyzing the problem, designing the system, and constructing the program. An object is a software package that contains both data and a collection of related structures and procedures. Since it contains both data and a collection of structures and procedures, it can be visualized as a self-sufficient component that does not require other additional structures, procedures or data to perform its specific task. OOP, therefore, views a computer program as a collection of largely autonomous components, called objects, each of which is responsible for a specific task. This concept of packaging data, structures, and procedures together in one component or module is called encapsulation.

In general, OOP components are reusable software modules which present an interface that conforms to an object model and which are accessed at run-time through a component integration architecture. A component integration architecture is a set of architecture mechanisms which allow software modules in different process spaces to utilize each others capabilities or functions. This is generally done by assuming a common component object model on which to build the architecture. It is worthwhile to differentiate between an object and a class of objects at this point. An object is a single instance of the class of objects, which is often just called a class. A class of objects can be viewed as a blueprint, from which many objects can be formed.

OOP allows the programmer to create an object that is a part of another object. For example, the object representing a piston engine is said to have a composition-relationship with the object representing a piston. In reality, a piston engine comprises a piston, valves and many other components; the fact that a piston is an element of a piston engine can be logically and semantically represented in OOP by two objects.

OOP also allows creation of an object that "depends from" another object. If there are two objects, one representing a piston engine and the other representing a piston engine wherein the piston is made of ceramic, then the relationship between the two objects is not that of composition. A ceramic piston engine does not make up a piston engine. Rather it is merely one kind of piston engine that has one more limitation than the piston engine; its piston is made of ceramic. In this case, the object representing the ceramic piston engine is called a derived object, and it inherits all of the aspects of the object representing the piston engine and adds further limitation or detail to it. The object representing the ceramic piston engine "depends from" the object representing the piston engine. The relationship between these objects is called inheritance.

When the object or class representing the ceramic piston engine inherits all of the aspects of the objects representing the piston engine, it inherits the thermal

characteristics of a standard piston defined in the piston engine class. However, the ceramic piston engine object overrides these ceramic specific thermal characteristics, which are typically different from those associated with a metal piston. It skips over the original and uses new functions related to ceramic pistons. Different kinds of piston engines have different characteristics, but may have the same underlying functions associated with it (e.g., how many pistons in the engine, ignition sequences, lubrication, etc.). To access each of these functions in any piston engine object, a programmer would call the same functions with the same names, but each type of piston engine may have different/overriding implementations of functions behind the same name. This ability to hide different implementations of a function behind the same name is called polymorphism and it greatly simplifies communication among objects.

With the concepts of composition-relationship, encapsulation, inheritance and polymorphism, an object can represent just about anything in the real world. In fact, our logical perception of the reality is the only limit on determining the kinds of things that can become objects in object-oriented software. Some typical categories are as follows:

- Objects can represent physical objects, such as automobiles in a traffic-flow simulation, electrical components in a circuit-design program, countries in an economics model, or aircraft in an air-traffic-control system.
- Objects can represent elements of the computer-user environment such as windows, menus or graphics objects.
- An object can represent an inventory, such as a personnel file or a table of the latitudes and longitudes of cities.
- An object can represent user-defined data types such as time, angles, and complex numbers, or points on the plane.

With this enormous capability of an object to represent just about any logically separable matters, OOP allows the software developer to design and implement a computer program that is a model of some aspects of reality, whether that reality is a

physical entity, a process, a system, or a composition of matter. Since the object can represent anything, the software developer can create an object which can be used as a component in a larger software project in the future.

If 90% of a new OOP software program consists of proven, existing components made from preexisting reusable objects, then only the remaining 10% of the new software project has to be written and tested from scratch. Since 90% already came from an inventory of extensively tested reusable objects, the potential domain from which an error could originate is 10% of the program. As a result, OOP enables software developers to build objects out of other, previously built objects.

This process closely resembles complex machinery being built out of assemblies and sub-assemblies. OOP technology, therefore, makes software engineering more like hardware engineering in that software is built from existing components, which are available to the developer as objects. All this adds up to an improved quality of the software as well as an increased speed of its development.

Programming languages are beginning to fully support the OOP principles, such as encapsulation, inheritance, polymorphism, and composition-relationship. With the advent of the C++ language, many commercial software developers have embraced OOP. C++ is an OOP language that offers a fast, machine-executable code. Furthermore, C++ is suitable for both commercial-application and systems-programming projects. For now, C++ appears to be the most popular choice among many OOP programmers, but there is a host of other OOP languages, such as Smalltalk, Common Lisp Object System (CLOS), and Eiffel. Additionally, OOP capabilities are being added to more traditional popular computer programming languages such as Pascal.

The benefits of object classes can be summarized, as follows:

- Objects and their corresponding classes break down complex programming problems into many smaller, simpler problems.

- Encapsulation enforces data abstraction through the organization of data into small, independent objects that can communicate with each other. Encapsulation protects the data in an object from accidental damage, but allows other objects to interact with that data by calling the object's member functions and structures.
- Subclassing and inheritance make it possible to extend and modify objects through deriving new kinds of objects from the standard classes available in the system. Thus, new capabilities are created without having to start from scratch.
- Polymorphism and multiple inheritance make it possible for different programmers to mix and match characteristics of many different classes and create specialized objects that can still work with related objects in predictable ways.
- Class hierarchies and containment hierarchies provide a flexible mechanism for modeling real-world objects and the relationships among them.
- Libraries of reusable classes are useful in many situations, but they also have some limitations. For example:
- Complexity. In a complex system, the class hierarchies for related classes can become extremely confusing, with many dozens or even hundreds of classes.
- Flow of control. A program written with the aid of class libraries is still responsible for the flow of control (i.e., it must control the interactions among all the objects created from a particular library). The programmer has to decide which functions to call at what times for which kinds of objects.
- Duplication of effort. Although class libraries allow programmers to use and reuse many small pieces of code, each programmer puts those pieces together in a different way. Two different programmers can use the same set of class libraries to write two programs that do exactly the same thing but whose internal structure (i.e., design) may be quite different, depending on hundreds of small decisions each programmer makes along the way. Inevitably, similar pieces of code end up doing similar things in slightly different ways and do not work as well together as they should.

Class libraries are very flexible. As programs grow more complex, more programmers are forced to reinvent basic solutions to basic problems over and over

again. A relatively new extension of the class library concept is to have a framework of class libraries. This framework is more complex and consists of significant collections of collaborating classes that capture both the small scale patterns and major mechanisms that implement the common requirements and design in a specific application domain. They were first developed to free application programmers from the chores involved in displaying menus, windows, dialog boxes, and other standard user interface elements for personal computers.

Frameworks also represent a change in the way programmers think about the interaction between the code they write and code written by others. In the early days of procedural programming, the programmer called libraries provided by the operating system to perform certain tasks, but basically the program executed down the page from start to finish, and the programmer was solely responsible for the flow of control. This was appropriate for printing out paychecks, calculating a mathematical table, or solving other problems with a program that executed in just one way.

The development of graphical user interfaces began to turn this procedural programming arrangement inside out. These interfaces allow the user, rather than program logic, to drive the program and decide when certain actions should be performed. Today, most personal computer software accomplishes this by means of an event loop which monitors the mouse, keyboard, and other sources of external events and calls the appropriate parts of the programmer's code according to actions that the user performs. The programmer no longer determines the order in which events occur. Instead, a program is divided into separate pieces that are called at unpredictable times and in an unpredictable order. By relinquishing control in this way to users, the developer creates a program that is much easier to use. Nevertheless, individual pieces of the program written by the developer still call libraries provided by the operating system to accomplish certain tasks, and the programmer must still determine the flow of control within each piece after it's called by the event loop. Application code still "sits on top of" the system.

Even event loop programs require programmers to write a lot of code that should not need to be written separately for every application. The concept of an application framework carries the event loop concept further. Instead of dealing with all the nuts and bolts of constructing basic menus, windows, and dialog boxes and then making these things all work together, programmers using application frameworks start with working application code and basic user interface elements in place. Subsequently, they build from there by replacing some of the generic capabilities of the framework with the specific capabilities of the intended application.

Application frameworks reduce the total amount of code that a programmer has to write from scratch. However, because the framework is really a generic application that displays windows, supports copy and paste, and so on, the programmer can also relinquish control to a greater degree than event loop programs permit. The framework code takes care of almost all event handling and flow of control, and the programmer's code is called only when the framework needs it (e.g., to create or manipulate a proprietary data structure).

A programmer writing a framework program not only relinquishes control to the user (as is also true for event loop programs), but also relinquishes the detailed flow of control within the program to the framework. This approach allows the creation of more complex systems that work together in interesting ways, as opposed to isolated programs, having custom code, being created over and over again for similar problems.

Thus, as is explained above, a framework basically is a collection of cooperating classes that make up a reusable design solution for a given problem domain. It typically includes objects that provide default behavior (e.g., for menus and windows), and programmers use it by inheriting some of that default behavior and overriding other behavior so that the framework calls application code at the appropriate times.

There are three main differences between frameworks and class libraries:

- Behavior versus protocol. Class libraries are essentially collections of behaviors that one can call when one wants those individual behaviors in a program. A framework, on the other hand, provides not only behavior but also the protocol or set of rules that govern the ways in which behaviors can be combined, including rules for what a programmer is supposed to provide versus what the framework provides.

- Call versus override. With a class library, the code the programmer instantiates objects and calls their member functions. It's possible to instantiate and call objects in the same way with a framework (i.e., to treat the framework as a class library), but to take full advantage of a framework's reusable design, a programmer typically writes code that overrides and is called by the framework. The framework manages the flow of control among its objects. Writing a program involves dividing responsibilities among the various pieces of software that are called by the framework rather than specifying how the different pieces should work together.

- Implementation versus design. With class libraries, programmers reuse only implementations, whereas with frameworks, they reuse design. A framework embodies the way a family of related programs or pieces of software work. It represents a generic design solution that can be adapted to a variety of specific problems in a given domain. For example, a single framework can embody the way a user interface works, even though two different user interfaces created with the same framework might solve quite different interface problems.

Thus, through the development of frameworks for solutions to various problems and programming tasks, significant reductions in the design and development effort for software can be achieved. A preferred embodiment of the invention utilizes HyperText Markup Language (HTML) to implement documents on the Internet together with a general-purpose secure communication protocol for a transport medium between the client and the Newco. HTTP or other protocols could be readily substituted for HTML without undue experimentation. Information on these products is available in T. Berners-Lee, D. Connoly, "RFC 1866: Hypertext Markup Language - 2.0" (Nov. 1995); and R. Fielding, H, Frystyk, T. Berners-Lee, J. Gettys

and J.C. Mogul, "Hypertext Transfer Protocol -- HTTP/1.1: HTTP Working Group Internet Draft" (May 2, 1996). HTML is a simple data format used to create hypertext documents that are portable from one platform to another. HTML documents are SGML documents with generic semantics that are appropriate for representing information from a wide range of domains. HTML has been in use by the World-Wide Web global information initiative since 1990. HTML is an application of ISO Standard 8879; 1986 Information Processing Text and Office Systems; Standard Generalized Markup Language (SGML).

To date, Web development tools have been limited in their ability to create dynamic Web applications which span from client to server and interoperate with existing computing resources. Until recently, HTML has been the dominant technology used in development of Web-based solutions. However, HTML has proven to be inadequate in the following areas:

- Poor performance;
- Restricted user interface capabilities;
- Can only produce static Web pages;
- Lack of interoperability with existing applications and data; and
- Inability to scale.

Sun Microsystems's Java language solves many of the client-side problems by:

- Improving performance on the client side;
- Enabling the creation of dynamic, real-time Web applications; and
- Providing the ability to create a wide variety of user interface components.

With Java, developers can create robust User Interface (UI) components. Custom "widgets" (e.g., real-time stock tickers, animated icons, etc.) can be created, and client-side performance is improved. Unlike HTML, Java supports the notion of client-side validation, offloading appropriate processing onto the client for improved performance. Dynamic, real-time Web pages can be created. Using the above-mentioned custom UI components, dynamic Web pages can also be created.

Sun's Java language has emerged as an industry-recognized language for "programming the Internet." Sun defines Java as: "a simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high-performance, multithreaded, dynamic, buzzword-compliant, general-purpose programming language. Java supports programming for the Internet in the form of platform-independent Java applets." Java applets are small, specialized applications that comply with Sun's Java Application Programming Interface (API) allowing developers to add "interactive content" to Web documents (e.g., simple animations, page adornments, basic games, etc.). Applets execute within a Java-compatible browser (e.g., Netscape Navigator) by copying code from the server to client. From a language standpoint, Java's core feature set is based on C++. Sun's Java literature states that Java is basically, "C++ with extensions from Objective C for more dynamic method resolution."

Another technology that provides similar function to JAVA is provided by Microsoft and ActiveX Technologies, to give developers and Web designers wherewithal to build dynamic content for the Internet and personal computers. ActiveX includes tools for developing animation, 3-D virtual reality, video and other multimedia content. The tools use Internet standards, work on multiple platforms, and are being supported by over 100 companies. The group's building blocks are called ActiveX Controls, small, fast components that enable developers to embed parts of software in hypertext markup language (HTML) pages. ActiveX Controls work with a variety of programming languages including Microsoft Visual C++, Borland Delphi, Microsoft Visual Basic programming system and, in the future, Microsoft's development tool for Java, code named "Jakarta." ActiveX Technologies also includes ActiveX Server Framework, allowing developers to create server applications. One of ordinary skill in the art readily recognizes that ActiveX could be substituted for JAVA without undue experimentation to practice the invention.

Figure **2** illustrates a flowchart for a process **200** for embedding keywords in video (digital video, in particular) in accordance with an embodiment of the present

invention. Multimedia content comprising one or more digital video images is displayed in operation **202**. One or more keywords are associated with at least one of the video images in operation **204**. In operation **206** a network is utilized to retrieve information relating to the keywords, by use of a search engine, for example, that executes a keyword search for one or more sites in the network that relate to the keyword. The retrieved information relating to the keyword is then displayed in operation **208**. This retrieved information may be displayed in a summary form that summarizes the sites found during the search relating to the keyword.

In an embodiment of the present invention, the keywords associated with the video images of the multimedia content may also be displayed. As an option, the keywords may be displayed adjacent to their associated video images. For example, the keyword "Nike" may be positioned adjacent a video image of a pair of Nike brand shoes.

To extend this further, the meta language may even include mapping coordinates on the screen that apply to a keyword. Therefore, in such an aspect, keywords are associated with areas on the screen so if the user uses an on-screen pointing device and points at the shoes the keyword "Nike" would appear since that keyword is set for those coordinates near the actors feet. The use of video viewing equipment that allows temporary pause and panning on the screen may further aid in the viewing of such content. For commercials being viewed, for example, the keyword may be the product name which the user may bookmark and search at that point or later in time as desired by the user. In such an illustrative embodiment, the user may be able to view a program (i.e., content) uninterrupted by commercials. While viewing the program, the user has the ability to "bookmark" scenes or keywords that are available at a particular point in time during the program. In our "Nike" example, if the user is watching a program and likes the Nike brand shoes the actor in the program is wearing, "Nike" could potentially be one of the keywords that is seen by the user. The user may then go back after "bookmaking" and when the show is over and perform a lookup on the word "Nike" to find out the location of the closest store (with respect to the user) that sells the particular shoes, information about them etc.

In one embodiment of the present invention, the keyword may comprise a numerical tag or code. In such an embodiment, the tag number/code may be submitted to a pre-processing engine to get a full list of information associated with the tag number/code. In an illustrative example, a keyword may relate to a promotional offer having a numerical tag/code such as, for example, PO123456. This number, PO123456 may then be submitted so that an entire list of related information is provided. In another illustrative example, as similar system may be used for home shopping television channels. In this example, a number is typically displayed on the television screen relating to the product being featured for sale at that time on the shopping channel. This number is the code for that particular product being sold. In this example, the user may either "bookmark" this number or utilize the network to submit the tag to the preprocessing engine so that information relating to this product is recalled. Thus in this example, the complete set of information is retrieved upon the user's request while only the number tags/keyword is broadcasted with the video stream.

In another embodiment of the present invention, one of the displayed video images of the content may be selected to thereby display the keywords associated with the selected video image. In such an embodiment, this may be accomplished by displaying a pointer and then permitting a user to position the pointer over the displayed image to select the particular video image to thereby display the associated keywords. As a further option, the user may also be able to select one of the displayed keywords with the pointer to execute a search for information relating to the selected keyword utilizing the network.

In a further embodiment of the present invention, the retrieved information may also be indexed based on the associated keywords. In yet another embodiment of the present invention, a user profile of a user may be obtained so that the retrieved information may include information related to the user profile of the user. For example, the user profile may include information relating to the location/residence of the user, the search may take into account this information so that the retrieved

information is from locations proximate to the user's location/residence. Returning to our Nike example, the user profile may provide information as to where the user lives and thereby allow the present invention to automatically provide the stores in the user's area that sell Nike brand shoes.

In an aspect of the present invention, the video images of the multimedia content may be carried/transmitted in a first channel and the keywords may be carried/transmitted in a second channel. In another aspect of the present invention, content may utilize the Advanced Television Enhancement Forum (ATVEF) Content Specifications. In such an aspect, the present invention may be integrated with ATVEF to be an extension to ATVEF where ATVEF is used as the secondary stream to the video that the keywords are carried in.

By using keywords, a very wide search of a network may be conducted than with simply a uniform resource locator (ULR). For example, in the present invention, a keyword search may include queries of where to get the product and more information about it. In addition, a Portal Language/Access may be included with the ability to display, synchronize and control display of Internet-based assets with digital video.

Instead of adding PC- based content into video stream, a common meta-language may be defined for access to data from a remote server. This meta language may be used for providing coordinates for hot spots. For example, with each keyword, a category/classification code may also be included as part of the meta language to assist in the search process. As an illustration, the codes could be: Clothing, Travel, Electronics, Food, and Medicine for example. These category/classification codes may be used to help searching for obscure products available through the Internet.

A benefit from this is that the present invention may help to greatly reduce the expense of trying to constantly create and insert data to video stream (whether it be optical disk technology such as digital versatile disk (DVD) or broadcast technology). As an illustrative example, PBS has a hard enough time creating the

TV show, let alone additional web based content. With the present invention, content online that can be displayed is indexed based on pre-defined criteria/keywords. The keywords are added to the video stream. The key words are also then used by a server to serve up/aggregate information from several locations. This way, PBS does not need to do new content creation, but instead is able to display relevant information found all over the web.

Keywords can link to search engines to applications for DVD and broadcast video. Additionally, weights may be applied to where searches can occur. The search results may then be displayed in a weighted ordered list. Further, user profiles and native languages may further tailor the server search.

In yet another aspect of the present invention, it should also be understood that the information that is searched may be stored in a media such as an optical storage media such as DVD Video.

In yet even a further embodiment of the present invention, the secondary keyword stream does not need to be synchronized nor accompany the video in real-time. By using in the case of television a replacement for the keyword can be the date, time, channel, cable provider of the program and this information can be submitted to a local or on-line database that then displays the keywords or requested information to the user so then they can browse the topic further. For the case of DVD-Video the keyword can be replaced by a unique disc identifier, Title, Chapter, Time and this information is submit into a local or on-line database to retrieve the requested information. Thus in this embodiment, the secondary stream may not need to be a stream synchronized with the video or a even a stream at all but just simply a separate database indexed by the unique video identifier. These unique video identifiers can either be stored in a table for future use in the case the user wants to bookmark the video or be used immediately at the users request.

SEARCH ENGINE

In one aspect of the present invention, a search engine may be used to retrieve information relating to the keywords utilizing the network as set forth in operation **206** of Figure **2**. Information is available from a variety of information resources. For example, the user may acquire information from the World Wide Web or via broadcasts, such as those sent via satellite and cables, that include information regarding the broadcast that allows the displaying of information. In an exemplary embodiment of the present invention, a search engine or search tool as set forth below may be utilized to execute the keyword search.

In such an exemplary embodiment, an integrated search engine may be provided for specifying and searching a variety of information resources. In one embodiment, the search engine may be used for searching broadcast information and Internet information using a single user-initiated search. The search criteria may be saved as a filter, which can be executed at a later time. Results of the search may also be presented to the user. The user can then display available Web sites and/or an electronic program guide (EPG) containing program information that meets the search criteria. Via the EPG, broadcasts can be selected and displayed on the display. Thus, the user can access broadcast information and Internet information on the same search topic and criteria without performing multiple searches or recreating the search criteria.

Sub A'

Figure **3** is a block diagram of a system that operates in accordance with an embodiment of the present invention. A variety of systems may be used. For example, a multimedia computer, such as the Sony PC manufactured by Sony Corporation may be utilized. The system **300** typically includes a central processing unit (CPU) **330**, memory **335**, input/output circuitry **325**, as well as other circuitry and components that are well known to those skilled in the art. The system **300** outputs information to a display **320** and, may also provide audio through speakers **326**. The information may be received through receiver **305**. Receiver **305** in one embodiment is a satellite receiver for receiving satellite transmissions of broadcasts

and programming information through antenna **306**. Using the programming information received through receiver **305**, the system **300** can generate an electronic program guide (EPG) on the display **320**. As will be described below, the EPG can be modified or filtered according to the searching performed by the user using the search engine described herein.

The user can provide input to the system **300** through a user input device **315** which may include a keyboard, mouse, remote control or other input device. The system **300** further has access to the Internet through Internet access **310**, and also can access previously accessed and stored web pages. Using this access mechanism, which may be an Internet provider or other connection to the Internet, the user can search for external information including information available on the World Wide Web and previously broadcasted Web pages. It is readily apparent that the system is not limited to Internet access and can access a variety of external or internal resources including third party databases.

An overview of the search engine is illustrated in the flow diagram of Figure **4**. The search engine includes query tools for specifying and selecting the filter elements used to perform the search. The user can select the information sources to be searched, such as the World Wide Web and electronic program guide (EPG) information. In the present embodiment, the World Wide Web and EPG information are accessed; however, it is readily apparent that the resources can be expanded to include other resources, and furthermore, that one, some, or all of the resources can be selected for searching. The user can also invoke commands to perform a search, stop a search, import information for performing the search, as well as maintaining logs of searches performed for subsequent references. Furthermore, the user can select what is to be searched and displayed, such that only web information is displayed, EPG information is only displayed, or all information is displayed.

Referring to Figure **4**, a user, using a search engine window **402** can establish the topics that form elements of a filter **404** that is input to a search engine **406**. The search engine **406** interacts with the different information resources, e.g., internet

A7 cancel

412, cable broadcast **410** and satellite broadcast **408**, to generate a result set **414** of information. This set **414** is applied to the EPG **416** to modify the EPG **416** to display or highlight those programs that meet the filter requirements. These results may then be displayed in the display **422**.

In addition, the user can maintain filter logs that reflect the filter terms used to perform the search. These logs can be selected, such that the search can be re-performed at a later time. The first step in the process is the creation of a filter to be used. This process is described with reference to Figure **5**. Text strings **501** are entered or selected for the topic list which indicate the topic or terms to be used to perform the search. In block **502**, the text can be entered by typing in information, or importing information from the EPG **504**, such as the current title of a program currently being broadcasted on the user's display. Logical operators can be used **508** to combine multiple terms. An existing filter can be used **510** by selecting existing filters from the filter log **512**. In addition, filters or search terms can be acquired from information associated with a broadcast **514**. For example, information such as broadcast categories **515** (news, sports, drama, etc.), cost **516**, rating **520**, length **522**, start time **524**, and end time **526** are examples of parameters supplied by the broadcast system for generation of an electronic program guide. This information can be used to generate the filters used to perform the search. All or some of these filters and terms can be used. The search is flexible to select one or a plurality of information resources. In the present embodiment, the user can select **528** to search the World Wide Web **530** or an electronic program guide **532**. It is readily apparent to one skilled in the art that other resources may be used.

Once the filter is created, the search may be activated. The process is illustrated with respect to Figure **6**. Commands **601** are used to specify certain parameters. After a search is initiated **602** using the active filter specified **604**, the search mechanism conducts a search of the World Wide Web **606**, and the EPG **608**. At any time the search may be stopped **610**; the filters added to the filter log **611**, the present filter delete from the log **614** or edited **616**, resulting in an updated filter log **618**. Using the filter specified, the system automatically generates the query to perform the

search on the web and/or on the EPG. This can be performed a number of ways recognized by those skilled in the art. For example, a script can be generated that executes the sequence of commands needed to access the web and perform the search using existing search engines or a specially created search engine. Similarly, the search is performed on the EPG using a search engine. The search engine may be a simple text search engine or database search engine, or a tool specifically written for searching the EPG.

Once the search has been performed, the results are presented to the user. This is illustrated by the exemplary display shown in Figure **7**. Referring to Figure **7**, results can be formatted a number of ways. In the present embodiment, the result **704** of any web searches **705** are presented in an frame **710** on the display. For example, if multiple web sites meet the search criteria, the user may be presented a listing of web sites with the ability to move a cursor over to a web site URL and select the URL to bring up the particular web site. Alternately, a first web site can be automatically brought to the user's display or multiple web sites can be displayed, and the user can go forward or back across the multiple sites inside the window.

The results of the search performed on the electronic program guide **715** are displayed a variety of ways. For example, the EPG is modified to only display those programs that meet the search criteria. Alternatively, the areas of the EPG corresponding to programs that meet the criteria are highlighted by a different color. Furthermore, in one embodiment, the user is able to change the current broadcast **725** to one of the programs currently broadcast that meet the search criteria. For example, this might be done by selecting a program from the modified EPG. Selection maybe achieved a variety of ways. For example, the user may indicate selection by using a remote control to enter the station number ID or by moving the cursor to point to the desired program. The system then responds by tuning to the program selected (the program being one of the programs that meets the search criteria).

In an alternative environment, the search is initiated in the background by the

selection of a program in the EPG. Preferably, the filter elements of a selected program, such as a particular broadcasted program on the user's desktop display, are determined from selected program elements of the EPG. For example, providers of satellite broadcasts provide electronic program guide streams from which the receiver devices can generate electronic program guides visible to the user. This information typically includes the title, abstract of the program, duration of the program, time of broadcast, and lead actors in the program. Upon selection of a particular broadcast to view, a background search can automatically be initiated using all or some of the parameters of the program element information provided with the program. For example, if the broadcast is a movie called "X", a search initiated by title could bring up the web site about the "X" movie. Alternately, if the broadcast is a show directed to the subject of whales, for example, a search can be initiated based on the abstract on the topic of whales, and web sites directed to that topic would automatically be provided to the user. Thus, the user would automatically receive information on a subject of interest from a variety of resources.

In either of the environments discussed above, the information associated with a broadcast can be more than just a sequence of keywords. Keywords can be combined with logical syntactic operators such as AND, OR and NOT to produce Boolean combinations of search terms and to provide a more intelligent query. For example, a popular search engine is the one provided by the Alta Vista site WWW.altavista.digital.com. Either the simple query or advanced query syntax as documented at this site may be used. Other query syntax may be used.

Additionally, a complete search query can be provided in association with a broadcast. That is, a string of keywords combined with operators can be included in the EPG associated with a broadcast, included in the vertical blank interval of a broadcast signal itself, included in an Internet server "push" of data to the system of the present invention in association with a broadcast program, etc. That is, the search query can be formed at the content-provider end rather than having the system at the user end construct the query.

As another refinement, search results can be provided by the content-provider so that the receiving user system does not have to perform a search. This last approach has the advantage of eliminating the unrelated information that may turn up from an Internet search but has the drawback that a large amount of information in the form of URL information must be transmitted. Still, where the number of URLs transmitted is small, this last approach may be the most efficient.

In one embodiment, this search is performed in the background so as not to disturb foreground processes, such as display of a broadcast or video. If the search identifies related web sites, for example, a discreet animated alert is provided to the user, for example, in the user's tools area, enabling the user to selectively bring up the related web sites by selecting the alert. If the user selects to view the web sites, the web sites are then displayed in the HTML window provided on the user's desktop display.

RANKING

In yet another aspect of the present invention, search results may be ranked utilizing an exemplary information retrieval and ranking system as set for below. In general, an information retrieval (IR) system is a computer-based system for locating, from an on-line source database or other collection, documents that are relevant to a user's input query. Until recently, most commercial IR systems, such as DIALOG.RTM. or LEXIS.RTM., used Boolean search technology. In a Boolean search system, users must express their queries using the Boolean operators AND, OR, and NOT, and the system retrieves just those documents that exactly match the query criteria. Typically, there is no score or other indication of how well each document satisfies the user's information need.

However, after years of research demonstrating the superiority of relevance-ranking, commercial systems began to offer this capability. Today millions of people use IR systems that employ relevance-ranking, also known as ranked searching, which is based on the "vector space model." In a relevance-ranked search system, users can

simply type an unrestricted list of words, even a "natural-language" sentence, as their query. The system then does a partial matching computation and assigns a score to every document indicating how well it matches the user's interest. Documents are then presented to the user in order, from the best matching to the least matching. Relevance-ranking is described in Salton, et al., Introduction To Modern Information Retrieval, McGraw-Hill Book Co., New York (1983). Relevance-ranking IR systems are commonly used to access information on the Internet, through systems based on the WAIS (Wide Area Information Servers) protocol or through a variety of commercial World Wide Web indexing service such as Lycos, InfoSeek, Excite, or Alta Vista. Relevance-ranking is also used in commercial information management tools such as AppleSearch, Lotus Notes and XSoft Visual Recall for searching databases or collections from individual or shared personal computers.

Relevance-ranking systems work as follows. In relevance-ranking, each word in every document of a collection is first assigned a weight indicating the importance of the word in distinguishing the document from other documents in the collection. The weight of the word may be a function of several components: (1) a local frequency statistic (e.g., how many times the word occurs in the document); (2) a global frequency statistic (e.g., how many times the word occurs in the entire collection of documents); (3) the DF measure (how many documents in the collection contain the word); and (4) a length normalization statistic (e.g., how many total words are in the document).

The following example demonstrates one possible term-weighting scheme for a relevance-ranking system. First, assume that a collection contains one-hundred (100) documents with one particular document containing only the text "the dog bit the cat." Assume further that the word "the" occurs in all 100 documents while the word "dog" occurs in five (5) documents and the word "cat" occurs in two (2) documents. Here, we use Term Frequency (TF), the number of times the word occurs in a particular document, as our local frequency statistic:

term=dog, TF=1,

term=the, TF=2,

term=cat, TF=1.

Here, we use DF as our global statistic:

term=dog, DF=5/100,

term=the, DF=100/100,

term=cat, DF=2/100,

where DF=number of documents containing the term total number of documents.

The inverses of DF (IDF) are calculated as follows:

term=dog, IDF=100/5=20,

term=the, IDF=100/100=1,

term=cat, IDF=100/2=50.

For this example, we will not use a length normalization statistic. Thus the final weights of each term using TF.times.IDF are as follows:

term=dog, TF.times.IDF=1.times.20=20,

term=the, TF.times.IDF=2.times.1=2,

term=cat, TF.times.IDF=1.times.50=50.

This list of weighted terms serves as the vector that represents the document. Note that terms found in more documents (such as "the") have lower weights than terms found in fewer documents (such as "cat"), even if they occur more frequently within the given document.

Every document in the collection is then assigned a vector of weights, based on various weighting methods such as TF.times.IDF weighting and weighting that takes TF.times.IDF and a length normalization statistic into account. After a query is entered, the query is converted into a vector. A similarity function is used to compare how well the query vector matches each document vector. This produces a score for each document indicating how well it satisfies the user's request. One such similarity function is obtained by computing the inner product of the query vector and the document vector. Another similarity function computes the cosine of the angle between the two vectors. Based on relevance-ranking, each document score is calculated and the retrieved documents are then outputted sequentially from the one with the highest score to the one with the lowest score.

A study performed by D. E. Rose and D. R. Cutting on an experimental information retrieval system by Apple Computer, Inc. of Cupertino, Calif. shows that casual users of IR systems prefer to issue short queries. During a four-week period from December 1995 to January 1996, over 50% of the 10,044 queries issued by at least 4,686 users in Apple's system contained only a single word, and no query was longer than 12 words. The mean query length was 1.76 words. A subsequent study performed by Rose and Cutting shows that out of 10,000 queries issued in Apple's system, over 53% were single-word queries and 94% were queries of three words or less. Similar results were obtained for queries placed in systems by Excite and the THOMAS system provided by the federal government. Rose, Daniel E. and Cutting, Douglass R., Ranking for Usability: Enhanced Retrieval for Short Queries, (submitted for publication, September 1996). Other studies have confirmed the preference of casual users for issuing short queries. Hearst, Marti A., Improving Full-Text Precision On Short Queries Using Simple Constraints, Fifth annual Symposium on Document Analysis and Information Retrieval, pp. 217-225 (1996).

The interfaces of the major Internet search services also encourage queries having few terms. The four well-known World Wide Web searching services (Lycos, InfoSeek, Excite, and AltaVista) present users with an entry field that accepts less than one line of text.

The statistical methods that provide relevance-ranking, such as "TF.times.IDF weighting" with the cosine similarity metric, attempt to "reward" documents that are well-characterized by each query term. In practice, this means that a document that has a very high value for some of the query terms may be ranked higher than a document that has a lower value for more of the query terms. Relevance-ranking algorithms are intended to achieve this outcome. However, users sometimes find that for short queries submitted to relevance-ranking IR systems, the users' goal of obtaining the most useful ordering of search results, from the most relevant document to the least relevant document, is not attained. Existing relevance-ranking algorithms may, in some circumstances when a query is short, assign higher scores to certain documents with low overlaps than to other documents with high overlaps. Overlap is determined by the number of terms common between the query and the document. This problem is exemplified in Figure **8**, which is a table partially showing the results of a short query entered into the Apple Developer web site. The query term entered by a user was "express modem," whereby the user probably intended to retrieve documents about the Apple Computer product by that name. The search results included 103 documents, and the documents with the top ten relevance scores are shown in Figure **8**. Column I shows the ranking of the search results based on relevance scores indicated by the symbol *. Column II shows the titles of the retrieved documents, and column III identifies which terms in the query were responsible for the documents being retrieved. The highest scoring document contained only the term "modem," as shown in row (a). This document discussed modems in general, without mentioning the term "express modem." The second highest scoring document contained only the term "express" (as shown in row (b)) and was not relevant to modems. The third highest scoring document did discuss the "express modem" product, as shown in row (c).

Figure **9** shows the method used by the prior art to produce the results described above with reference to Figure **8**. The method starts with operation **950**, where a query defining the search criteria is issued to a database or other information retrieval system. Next, operation **955** identifies a set of documents that meet the criteria defined in the query. Finally, operation **960** assigns a relevancy ranking to each of the documents in the identified set using conventional relevancy ranking algorithms discussed above. A possible solution to the short query ranking problem discussed above is to use queries based on Boolean search technology. However, the Boolean approach sacrifices the benefits of relevance-ranking, while research has shown that most casual users do not understand Boolean logic and have difficulty in using Boolean IR systems. Attempts have been made to ease user problems with Boolean systems with solutions that blend Boolean and relevance-ranking. Noreault T., Koll, M., and McGill, M. J., Automatic Ranked Output From Boolean Searches In SIRE, Journal of the American Society For Information Science, Vol. 26, No. 6, pp. 333-39 (1977); and Salton, G, Fox, E. A., and Wu, H., Extended Boolean Information Retrieval, Communications of the ACM, Vol. 26, No. 12, pp. 1022-1036 (1983). However, the above approaches combine Boolean and relevance-ranking, and consequently users are still required to express their queries as Boolean expressions if they wish to take advantage of the Boolean constraints.

G. Salton and C. Buckley have suggested that the statistical weighting of a short query should differ from the statistical weighting of a long query. Salton, G. and Buckley, C., Term-Weighting Approaches In Automatic Text Retrieval, Information Processing & Management, Vol. 24, No. 5, pp. 513-523 (1988).

One study that notes the short query problem is by Hearst, Marti A., Improving Full-Text Precision On Short Queries Using Simple Constraints, Fifth Annual Symposium on Document Analysis and Information Retrieval, pp. 217-225 (1996). However, this approach limits ranking within the confines of the Boolean search, and only if users input their query in a prescribed way.

In yet another aspect of the present invention, a further process for retrieving information in response to a query by a user may be utilized. This process includes the steps of receiving a signal s having a value corresponding to a relevance-ranking algorithm score of a retrieved document, receiving a signal q having a value corresponding to the number of words in the query and a signal v having a value corresponding to the coordination level of the retrieved document and query (i.e., the degree of overlap between the document terms and the query terms), and generating an adjusted score s1 dependent on the signal s, the signal q and the signal v. The adjusted score s1 takes the coordination level into account for small values of q and gradually decreases the importance of the coordination level as q increases. The system of this invention includes a computer-based system for carrying out the method of this invention.

In closer detail, Figure **10** is a flowchart describing the adjusted relevancy ranking method. The method starts with operation **1050**, where a query defining search criteria is issued to a database or other information retrieval system. Next, operation **1055** identifies a set of documents that meet the criteria defined in the query. Operation **1060** calculates a relevancy ranking for a document in the identified set and assigns the ranking score to a variable "s". Operation **1065** then calculates the overlap, which, as described above, is the number of terms in the query that appear in the document. The overlap value is assigned to a variable "v". Next, operation **1070** calculates the number of terms in the query and assigns the calculated value to a variable "q". Operation **1075** obtains the value of .differential., which is used to adjust the impact of the adjustment on the relevancy ranking score. Finally, operation **1080** determines the adjusted relevancy ranking score using Eq. 1.

The adjusted relevance-ranking algorithm of Eq. (1) has been measured against the standard cosine ranking method using the TREC-4 test collections. In calculating the R-precision (i.e., the precision after R documents, where R is the total number of relevant documents for the query), Eq. (1) shows an improvement over the TF.times.IDF cosine ranking method by 21.3%, 10.4%, 11.9% and 7.9% for query

terms containing two, three, four, and all words respectively in the document.

BOOKMARKING

Familiar to most people, traditional bookmarks are those used to mark a page in a book to which the reader wants to later return. In the personal computer and World Wide Web environment, an analogous feature allows a typical net browser application running on the computer to "bookmark" web pages, i.e., select a button from a pull-down menu on the browser tool, allowing the user to store a URL associated with a website for rapid, one-step return access, without requiring the user to recall or re-enter the URL of that particular website. The following portion of the specification focuses on an illustrative aspect of the present invention for permitting the user to bookmark one or more points/places in a multimedia presentation or broadcast. In particular, a system may be provided for marking viewer-selected multimedia or television (TV) broadcast events by selecting a one or more broadcast events using a remote control, and storing a set of data associated with each selected broadcast events as an activity record (AR) in an activity table (AT). The activity table with the set of event identifiers is transmitted to an on-line database having information relating to TV program schedules, TV and Web advertisements information and related website hotlinks to thereby generate a set of associated network locations, such as websites and website hotlinks. The generated set of associated internet locations or website hotlinks can be used by the viewer for access to and display of the generated set of internet locations or websites associated with viewer selected broadcast events.

Figure **11** illustrates an embodiment of a low cost TV event marking system **1100** for marking viewer selected TV broadcast events so that associated information such as websites can be retrieved from an on-line service **1160**, such as the Internet, an intranet or other networks. In effect, this provides means to "bookmark" a televised event, marking that event for later recall, loosely analogous to placing a bookmark in a book to facilitate later recall. TV event marking system **1100** allows a viewer to

"bookmark" a set of selected TV events as they are broadcast, such as, but not limited to, a TV advertisement, a TV news broadcast, a TV educational or entertainment program, or a TV job training show. TV event marking system **1100** stores the set of selected events into computer memory so that on-line data associated with these events can be retrieved from a central database. In system **1100**, the viewer can mark the specified broadcast event by activating a select button **1115** on a remote control **1112**. In this example, select button **1115** is labeled "B" on remote control **1112** to denote "Bookmark". Each time the viewer activates select button **1115** to bookmark a particular broadcast event, an activity record (AR) entry comprises data describing the date, time and channel is stored into an electronic memory **1180**. It is envisioned that TV event marking system **1100** can also store an AR entry for each additional data relating to viewer preferences, such as for example, each time the channel is changed via remote control **1112**, and other remote control operations indicating viewing preferences. We refer to a list of AR entries as an activity table (AT) **1184** that is stored in electronic memory **1180**.

Sub A3 > Once the viewer has completed marking a selection of broadcast events, AT **1184** is stored into a network access device **1121**, whether in the resident memory inside network access device **1121** coupled to a TV tuner **1134**, or in an alternative embodiment, in the resident memory of a personal computing device **1120**. When the viewer is ready to browse the websites associated with the selected broadcast events, either network access device 21, or personal computing device **1120**, transmits activity table **1184** comprising the AR entries and also viewer identifying data, such as a particular demographic data, for example, the postal code of the viewer's location, via on-line service **1160** to a central database **1140**. Database **1140** comprises information compiled from various sources, such as TV advertisements schedules **1150** associated with various TV shows, TV show schedules **1152**, TV advertisers' websites **1162** and other websites topically related to broadcast content **1164**. AT **1184** is then used to determine which data in the database **1140** should be retrieved and presented to the viewer. For example, one of the AR entries in the AT might be (Sep. 1, 1999-19:30:32-CH7), indicating the date, time, and channel selected. This data, along with the viewer's regional information, is then compared to

A3 cancl

the TV advertisement schedule **1150** in database **1140** to determine the TV advertisements broadcast at the time of activating select button **1115**. Database **1140** then generates a custom list of data for the user which indicates bookmarks associated with the broadcast event. For example, this list of data could take the form of, but not limited to, a World Wide Web (WWW) page on the Internet. The viewer could then view these with a generic WWW browser.

In one embodiment of this aspect, the network access device may comprise a set-top box comprising a computer system coupled to a conventional TV tuner, or a specialized TV having computer processing capability (i.e., a PCTV), both having conventional network connection capabilities or other means for on-line access to the Internet or other networks. A CPU controls among other functions, a wireless interface, a custom command table, communications with external devices via an I/O interface. In the preferred embodiment, the AT is stored in electronic memory inside the network access device. When a Bookmark button is pressed, the remote control sends a wireless signal comprising a command to the CPU to store an AR entry into the AT inside the network access device, thereby "bookmarking" the broadcast event for later lookup.

As previously mentioned, with such an embodiment, the secondary keyword stream does not need to be synchronized nor accompany the video in real-time. By using in the case of television a replacement for the keyword can be the date, time, channel, cable provider of the program and this information can be submitted to a local or on-line database that then displays the keywords or requested information to the user so then they can browse the topic further. For the case of DVD-Video the keyword can be replaced by a unique disc identifier, Title, Chapter, Time and this information is submit into a local or on-line database to retrieve the requested information. Thus in this embodiment, the secondary stream may not need to be a stream synchronized with the video or a even a stream at all but just simply a separate database indexed by the unique video identifier. These unique video identifiers can either be stored in a table for future use in the case the user wants to bookmark the video or be used immediately at the users request.

Additionally, the remote control may include a network access button that interrupts a TV broadcast displayed on a TV and immediately display instead a selected associated website on the TV. In this example, the network button may be labeled "Go" to denote "Go to selected site". Each time the viewer activates this network button, a request to view a particular website is initiated. Aside from the new features described herein, the remote control may further comprise similar basic components and functions as in conventional remote controls, and thus it also provides the traditional operations of other conventional remote controls along with the event marking function buttons, such as provided by an event selection button, a network access button, and in an alternative embodiment, a download button and upload button.

In operation, whenever the viewer activates the event selection button "B" on keypad, this activation contemporaneously triggers the CPU in the network access device to concurrently query a real time clock for the current date and time, and an IR command table for the current channel, in order to generate an AR to which it adds a flag indicating a "bookmark". The resulting activity record having a "bookmark" flag is then stored into the AT and, as an example, comprising information representative of:

Date-Time-Channel-Bookmark:

Sep. 1, 1999-19:30:32-7-B

The TV event marking system may also be used to provide data of user's viewing patterns, interests and preferences by generating an AR entry for each time the viewer changes channels via the remote control. Whenever the viewer activates the change of channel button on the remote control, the remote control sends to the TV, the VCR, etc., an infrared signal to change the channel, and also signals the CPU in the network access device to query the real time clock circuit for the current date and time, and also the current channel register for the current channel information. The

resulting AR entry might comprise the following representative information:

Date-Time-Channel Change:

Sep. 1, 1999-19:30:32-CH7

Each change of channel by the viewer thereby produces a corresponding stored AR entry in AT **1182**, the collective data of various AT **1182** from each viewer can be used to evaluate viewer preferences and viewing patterns.

The number of AR entries stored in the AT is limited only by the available memory space in the memory storage or attached storage device. The AT may be organized in a first-in, first-out (FIFO) sequence. When the available memory is fill, the oldest ARs are deleted to accommodate the newest ARs. When the viewer wants to access the various websites associated with the selected broadcast events, the viewer activates network access button **1116** ("Go") which causes peripheral device **1121** to send the selected the AT to the database, whereupon the database will return to the network access device the network address of the selected websites. The network access device may then processes the network address for the selected website and retrieve it for the viewer. Thus the viewer can access selected websites with a single button.

In an alternative embodiment of network access device, the network access device need not be a set-top box, a PCTV, or a computer coupled to the TV for network access. In such an embodiment, AT may be first stored in a remote control device and later downloaded from remote control to a personal computing device, such as a standard personal computer that can communicate with remote control via a wireless interface and/or a standard I/O interface. In this embodiment, the personal computer has a network connection or other means for on-line access to the Internet and other such networks. The network interface for controlling the access to the network resides in the PC, thereby eliminating the need for a TV to be coupled to a settop

box or a personal computer.

Sub A4 > Figure **12** illustrates an automated custom program scheduling method using TV event marking system **1100** of Figure **11**. Automated custom program schedule method **1200** accesses on-line broadcast event listings in database **1140** to allow viewer to bookmark in advance selected scheduled broadcast events or websites for automated TV viewing. Automated custom scheduling method **1200** comprises a first operation **1202** of accessing database **1140** via network accessing device **1120** to view scheduled broadcast events. Then, in operation **1204**, viewer selects the set of broadcast events to be viewed. Once selection is completed, a corresponding custom schedule identifying selected the date, time and channel of all selected events is generated in operation **1206**. Then, in operation **1208**, the custom schedule is downloaded to custom command table in memory **1180** of network access device **1121**. The custom command table comprising a time-based command sequence is then executed by the CPU in the network access device **1121** in operation **1210** to instruct TV tuner **1134** to automatically change channels in a time sequence provided in the custom command table. It is envisioned that remote controls comprises bi-directional I/O port and thus can be remotely programmed by personal computer system or network access device.

INDEXING AND SEARCHING

Figure **13** is an overview of an illustrative embodiment of a system **1360** for computerized searching of data. The software system **1360** for computerized searching of data comprises at least one or more of the following programs: the Proximity Indexing Application Program **1362**, the Computer Search Program for Data Represented by Matrices (CSPDM **1366**) and the Graphical User Interface (GUI) Program. Proximity Indexing is a method of identifying relevant data by using statistical techniques and empirically developed algorithms. The Proximity Indexing Application Program **1362** is an application program which represents or indexes the database to a proper format to enable the Computer Search Program for Data

Represented by Matrices (CSPDM **1366**) to properly search a database. The Proximity Indexing Application Program **1362** can index data in a local database or a remote database.

After the Proximity Indexing Application Program indexes the database, the CSPDM application program can adequately search the database. The CSPDM program searches the database for objects according to instructions that the user enters into the Computer Processor via the input means. The CSPDM then retrieves the requested objects. The CSPDM either relays the objects and other information to the GUI program **1370** in order for the GUI program to display this information on the display, or the CSPDM sends display commands directly to the Computer Processor for display of this information. However, in the preferred embodiment, the CSPDM relays the objects and other commands to the GUI Program.

After the CSPDM has retrieved the objects, the Graphical User Interface (GUI) Program **1370**, which is a user interface program, causes the results of the search to be depicted on the display. The GUI Program enhances the display of the results of the search conducted by the CSPDM. The GUI Program, its method and operation, can be applied to other computer systems besides a system for computerized searching of data.

In one exemplary embodiment of the present invention, keywords may be indexed using the following process. Current computer search programs use a text-by-text analysis procedure (Boolean Search) to scan a database and retrieve items from a database. The user must input a string of text, and the computer evaluates this string of text. Then the computer retrieves items from the database that match the string of text. The two popular systems for computerized searching of data used in the legal profession are Westlaw.TM., a service sold by West Publishing Company, 50 W. Kellogg Blvd., P.O. Box 64526, St. Paul, Minn. 55164-0526, and Lexis.TM., a service sold by Mead Data Central, P.O. Box 933, Dayton, Ohio 45401.

Boolean searches retrieve exactly what the computer interprets the attorney to have

requested. If the attorney does not phrase his or her request in the exact manner in which the database represents the textual object, the Boolean search will not retrieve the desired textual object. Therefore, the researcher may effectively by denied access to significant textual objects that may be crucial to the project on which the researcher is working. A Boolean search also retrieves a significant amount of irrelevant textual objects. It should be noted that in the context of research, a textual object could be any type of written material. The term textual object is used to stress the fact that the present invention applies to all types of databases. The only requirement that a textual object must satisfy in order to be selected by a Boolean search program is that part of the textual object match the particular request of the researcher. Since the researcher cannot possibly know all of the groupings of text within all the textual objects in the database, the researcher is unable to phrase his request to only retrieve the textual objects that are relevant.

This aspect of the present invention may be used with an existing database by indexing the data and creating a numerical representation of the data. This indexing technique called proximity indexing generates a quick-reference of the relations, patterns, and similarity found among the data in the database. Using this proximity index, an efficient search for pools of data having a particular relation, pattern or characteristic can be effectuated. This relationship can then be graphically displayed.

As discussed above in Figure **13**, there are three main components to the invention; a data indexing applications program, a Computer Search Program for Data Represented by Matrices ("CSPDM"), and a user interface. Each component may be used individually. Various indexing application programs, CSPDMs, and user interface programs can be used in combination to achieve the desired results. The data indexing program indexes data into a more useful format. The CSPDM provides efficient computer search methods. The CSPDM includes multiple search subroutines. The user interface provides a user friendly method of interacting with the indexing and CSPDM programs. The user interface program allows for easy entry of commands and visual display of data via a graphical user interface.

The method which the invention uses to index textual objects in a database is called Proximity Indexing. This method can also be used to index objects located on a network. Proximity Indexing is a method of preparing data in a database for subsequent searching by advanced data searching programs. Proximity Indexing indexes the data by using statistical techniques and empirically developed algorithms. The resulting search by an advanced data searching program of the Proximity Indexed data is significantly more efficient and accurate than a simple Boolean search.

The Proximity Indexing Application Program indexes (or represents) the database in a more useful format to enable the Computer Search Program for Data Represented by Matrices (CSPDM) to efficiently search the database. The Proximity Indexing Application Program may include one or more of the following subroutines, an Extractor, a Patterner, and a Weaver. The Proximity Indexing Application Program indexes (or represents) data in a locally located database or remotely located database. The database can contain any type of data including text, alphanumeric, or graphical information.

In one aspect, the database may be located remotely from the Computer Processor and also contain some data in the form of textual objects. In this aspect, the Proximity Indexing Application Program indexes the textual objects by determining how each full textual object (e.g., whole judicial opinion, statute, etc.) relates to every other full textual object by using empirical data and statistical techniques. Once each full textual object is related to each other full textual object, the Proximity Indexing Application Program compares each paragraph of each full textual object with every other full textual object as described above. The Proximity Indexing Application Program then clusters related contiguous paragraphs into sections. Subsequently, the Proximity Indexing Application Program indexes each section and the CSPDM evaluates the indexed sections to determine which sections to retrieve from the database. Such organization and classification of all of the textual objects in the database before any given search commences significantly limits the irrelevant textual objects that the CSPDM program retrieves during the

subsequent search and allows retrieval of material based on its degree of relevancy.

In another aspect, the Proximity Indexing Application Program may include a link generation subroutine wherein direct and indirect relationships between or among data is used to generate a representation of the data. Generally, direct and indirect relationships in the database are identified as links and placed in a table. Again, this method of computerized research can be used for nearly any database including those containing non-textual material, graphical material, newspapers material, data on personal identification, data concerning police records, etc.

The remaining two programs are the CSPDM and the GUI Program. The CSPDM has seven subroutines that each search for different pools of objects. The GUI Program also has seven subroutines. Each CSPDM subroutine performs a different type of search. Each of the subroutines of the GUI uses the results of the corresponding subroutine of the CSPDM to create the proper display on the display. After the Proximity Indexing Application Program indexes a database, the CSPDM application program is used to search the indexed database. For example, the CSPDM program can either be located in memory that is remote from the Computer Processor or local to the Computer Processor. In addition, the CSPDM program can either be remote or local in relation to the database. The subroutines of the CSPDM may utilize the coefficients and other data created by the Proximity Indexing Application Program to facilitate its search. However, if the researcher does not have the particular object citation available, the researcher can perform a Boolean search to retrieve and organize a pool of objects. Alternatively, the researcher can subsequently search for related objects by using the Pool-Similarity Subroutine, the Pool-Paradigm Subroutine, the Pool-Importance Subroutine or the Pool-Paradigm-Similarity Subroutine as defined below.

If the researcher already has the citation of a particular object available, the researcher can search for related objects by utilizing the Cases-In Subroutine, Cases-After Subroutine or Similar-Cases Subroutine. The Cases-In Subroutine retrieves all of the objects from the database to which a selected object refers. In addition, the

subroutine determines the number of times the selected object refers to each retrieved object and other characteristics of each object, including its importance, and degree of relatedness to the selected object.

The Cases-After Subroutine retrieves all of the objects from the database that refer to the selected object. Also, the subroutine determines the number of times each retrieved object refers to the selected object and other characteristics of each object, including its importance, and degree of relatedness to the particular object to which it refers.

The Similar-Cases Subroutine determines the degree of similarity between the retrieved objects and the selected object. Similarity may be defined, in the context of legal cases, as the extent to which the two objects lie in the same lines of precedent or discuss the same legal topic or concept. Numerous other relationships may be used to define similarity.

In addition, for a textual object, if the researcher does not know of a particular textual object on which to base his or her search, the researcher may execute a Boolean word search. After a standard Boolean word search has been run, the researcher may run the Pool-Similarity Subroutine to retrieve information containing the degree of similarity between each textual object in the pool and a particular textual object selected by the user. Similarly, the Pool-Importance Subroutine can be used to determine the degree of importance (i.e., whether a judicial opinion is a Supreme Court opinion or a District Court opinion) and other characteristics of each textual object retrieved using the Boolean word search.

The Pool-Paradigm Subroutine calculates the geographic center in vector space of the pool of textual objects retrieved by the Boolean word search or other pool generating method. It then orders the retrieved textual objects by their degree of similarity to that center or "paradigm." The researcher can then evaluate this "typical textual object" and utilize it to help him or her find other relevant textual objects. In addition, the researcher can scan through neighboring "typical textual objects" to

evaluate legal subjects that are closely related to the subject of the researcher's search.

The Pool-Paradigm-Similarity Subroutine similarly creates a paradigm textual object from the retrieved textual objects. However, the subroutine calculates the similarity of all textual objects in the database to the paradigm textual object in addition to the similarity of the retrieved textual objects to the paradigm textual object.

After the CSPDM has retrieved the desired objects, the Graphical User Interface (GUI) Program may be used to display the results of the search on the display. In one embodiment, the GUI is a user interface program. The GUI Program contains three main subroutines: Cases-In Display Subroutine (CIDS), Cases-After Display Subroutine (CADS) and Similar-Cases Display Subroutine (SCDS). The main subroutines receive information from the corresponding subroutines Cases-In, Cases-After and Similar-Case s of the CSPDM. The GUI Program also contains four secondary subroutines: Pool-Similarity Display Subroutine ("PSDS"), Pool-Paradigm Display Subroutine ("PPDS"), Pool-Importance Display Subroutine ("PIDS"), and the Pool-Paradigm-Similarity Subroutine (PPSDS). The secondary subroutines also receive information from the corresponding subroutines Pool-Similarity Subroutine, Pool-Paradigm Subroutine, Pool-Importance Subroutine and the Pool-Paradigm Similarity Subroutine of the CSPDM.

The CIDS subroutine receives information gathered from the Cases-In Subroutine of the CSPDM. The CIDS subroutine displays user friendly active boxes and windows on the display which represent the textual objects retrieved from the database represented in Euclidean space. It can also use the boxes to represent objects retrieved from a network. Various active box formats and arranging of information within the boxes may be utilized. The display depicts the appropriate location of textual objects in Euclidean space on a coordinate means. An algorithm may be used to determine the appropriate location of the boxes. The coordinate means may have one or more axis. In one embodiment, the horizontal axis of the coordinate means may represent the time of textual object creation; the vertical axis could represent a

weighted combination of the number of sections in which that particular retrieved text is cited or discussed, its degree of importance, and its degree of similarity to the host textual object and the depth axis (Z-axis) represents the existence of data and length of the textual data or object.

The invention can also alter the background color of the window itself to communicate additional information graphically to the user. For example, if the horizontal axis represented time, then the invention could display the portion of the window containing objects occurring previous to the search object in one color and the portion containing the objects occurring after in another. Thus, the researcher can understand at a glance the relative position of his search target in relation to all the other objects related to it.

CIDS also enables the researcher to open up various active boxes on the display by entering a command into the computer processor with the input means. After entering the proper command, the active box transforms into a window displaying additional information about the selected textual object. These windows can be moved about the display and stacked on top or placed beside each other via the input means to facilitate viewing of multiple windows of information simultaneously. In one embodiment, the windows are automatically arranged by the computer system. Since the number of textual objects retrieved in a single search may exceed the amount which could be displayed simultaneously, the GUI Program enables the researcher to "zoom in" or "zoom out" to different scales of measurement on both the horizontal and vertical axis.

The CADS receives information gathered by the Cases-After Subroutine of the CSPDM. The CADS creates a display similar to the CIDS display. However, the active boxes representing the retrieved textual objects indicate which textual objects in the database refer to a selected textual object as opposed to which textual objects a selected textual object refers.

The SCDS receives information gathered by the Similar-Cases Subroutine of the

CSPDM. The SCDS causes a similar display on the display as the CIDS and the CADS except that the vertical axis indicates the degree of similarity between the retrieved textual objects and the selected textual object.

The GUI Program contains four secondary subroutines: Pool-Search Display Subroutine (PSDS), Pool-Paradigm Display Subroutine (PPDS), Pool-Importance Display Subroutine (PIDS) and the Pool-Paradigm-Similarity Display Subroutine (PPSDS). The PSDS receives the results gathered by the Pool-Search Subroutine of the CSPDM. The PPDS receives the results gathered by the Pool-Paradigm Subroutine of the CSPDM. The PIDS receives the results gathered by the Pool-Importance Subroutine of the CSPDM. The PPSDS receives the results gathered by the Pool-Paradigm-Similarity Subroutine of the CSPDM. The results of the PSDS, PPDS, PIDS and PPSDS are then displayed in a user friendly graphical manner similar to the results of the CIDS, CADS and SCDS. A researcher can access the PSDS, PIDS, PSDS or PPSDS from any of the three main or four secondary subroutines of the GUI to gather information corresponding to the active boxes that represent the pool of textual objects retrieved by the corresponding subroutine of the CSPDM.

By using the graphical display, the researcher can view immediately a visual representation of trends in the data (for example, trends developing in the law and current and past legal doctrines). In addition, the researcher can immediately identify important data or important precedent and which object serving as the precedent is most important to the project on which the researcher is working. This visual representation is a vast improvement over the current computerized research tools. Furthermore, the researcher using the present invention does not have to rely on the interpretation of another person to categorize different textual objects because the researcher can immediately visualize the legal trends and categories of law. In addition, new topic areas can be recognized without direct human intervention. The current research programs require a researcher to view objects in a database or to read through the actual text of a number of objects in order to determine which objects are important, interrelated, or most closely related to the topic at hand and

which ones are not.

This computerized system for researching data is also effective with any type of internal or global network application (see generally Figures **14** and **15**). As long as a network stores data and provides links between that data, this system can provide an effective and efficient system for indexing, searching, and displaying that data. For example, this system can be applied to the Internet and the World Wide Web. The World Wide Web is made up of numerous web sites which contain documents and internet or web pages. Documents are usually defined in the art as unique pieces of data, which includes text files, graphic files, audio files, and video files. A web page is usually a document with its own Universal Resource Locator (URL). URLs are the standardized addresses commonly used for web pages. Generally, web sites are a collection of web pages and documents. Web sites are usually identified by a home page, which may contain an overall starting point for the web site and a summary of what is to be found at the web site. Hyperjump links, or hyperlinks, is the name commonly given to the links which connect web pages, web sites, and documents on the web. Hyperlinks are electronic links which allow end users to jump to the specified web page or web site. The software code commonly used to create the majority of web pages containing text files is HyperText Markup Language (HTML). Other pages containing graphics, audio, video, and other resources may not be coded in HTML, but still will be connected by hyperlinks.

The Internet can be viewed as an immense collection of linked documents providing varied information to the public via an elaborate electronic distribution channel. In the past, the end user's ability to search, find, index, and navigate through relevant documents of interest has been primarily limited to word based queries which primarily rely on the target document's text indexing. Instead of relying on textual searching, this method and apparatus for indexing, searching, and displaying data analyzes hyperlinks which connect web pages to other web pages in order to help the end user to search, find, and navigate through the relevant documents of interest. This system analyzes hyperlinks using proximity indexing or clustering technology discussed previously. Once identified, the system displays the results in a variety of

ways and end users are able to navigate directly to the documents identified by this system's analyzation technology.

In one embodiment, this system uses a cluster link generation algorithm to search and identify closely associated documents located on the Internet in the same manner as described above. The system treats hyperlinks on the Web in the same manner as it treats links in a database, and it treats web pages on the Web in the same manner as it treats nodes in a database. Source links on the Web link a source node (or source web page) to a second node (or second web page). Influence links perform the same function in reverse. Direct links are the same as hyperlinks, which use URLs, in the World Wide Web, and they directly link one web page (or node) to another. Indirect links link two web pages or nodes through more than one path. A cluster link, for purposes of the Web, is any relationship between two web pages.

To begin the process, as shown generally in Figure **14**, a node is chosen **1400** for analysis. Next, the system accesses link data **1404** or "crawls" the source web page (or source node) looking for URLs which directly link the source web page to other web pages. Web crawling is a known technique in the art, performed by most World Wide Web search services, such as Yahoo (located at WWW.yahoo.com) or Alta Vista. Crawling is accomplished by the use of automated programs called robots or spiders, which analyze a web page for objects which provide URL links to other web pages or documents. The source node, whether it is a web page, the home page of a web site, or a document with no links, is a data document which may have been encoded in HTML or some other language. The encoded data document includes commands such as "insert picture here" or "begin a new paragraph" or "place a link here to another document" along with the normal text of the document. These coded commands are generally invisible to the end user, although many Web documents reveal text containing coded links to other documents in different colors. The system reads the coded HTML instructions to identify **1408** the coded links, which are direct links. There are many publicly known methods of identifying links from a coded document that one skilled in the art could employ to perform this function.

Figure **15** describes an aspect which executes **1412** the cluster link generator algorithm to generate direct and indirect links to find the set of candidate cluster links. After identifying **1408** all of the URLs referenced in the source web page, in the preferred embodiment, the cluster link generation algorithm retrieves a list of URLs and classifies them as the direct links to be analyzed. The cluster link generator traces the links to their destination nodes (a web site or web page) and performs a web crawl to retrieve a list of URLs referenced by the source nodes. The generator classifies the second set of nodes as being indirectly linked to the source node, and the links to these nodes are added to the list of candidate cluster links. In the more general method described in Figure **14**, the system identifies **1416** the links which have an indirect relationship and then displays **1420** the direct and indirect links.

Once a candidate cluster link set is identified, the generator assigns, weights to the candidate cluster links. The weight of each individual path or link is a function of the weight of the path to the previous node and the weight of the last link. In order to determine the weight of an implied link, the preferred formula, WC.sub.i+1 =min(WC.sub.i,D.sub.i+1 *W.sub.i+1) may be used. Following weighting, the generator sorts the set of candidate cluster links by weight, and a subset of these links (those links above a specified cut-off weight) are retained for display v3020 to the end user. In the preferred embodiment, the formula T=min(constant, 4*d), discussed before determines the optimal cut-off weight.

In another embodiment, the Proximity Indexing Application Program (Program) may organize and categorize the crawled links using the statistical techniques and empirically generated algorithms described earlier in this application. The Program treats URL addresses as citations and web pages as textual objects. The Program applies some or all of the eighteen pattern list to determine the relatedness of the web pages (or nodes) which are linked to the source web page (or node). The Program weighs the patterns by importance, giving one type of data document more importance than another type. For example, it may give more importance to a web site than to a single document which has no other links. The Program may use other

factors to weigh the data documents, such as the number of "hits" (visits by other end users to the site, a number which is available to web users) a data document receives in a specific time frame or the number of hyperlinks within a page. The Program then forms a matrix based on ordered pairs of documents, and the matrix calculations discussed before of this specification can be carried out. The Program generates a coefficient of similarity which will determine the relatedness of web pages to each other and to the source web page. The Program displays the most similar web pages to the user.

The preferred embodiment of the network application of this system uses the graphical user interface program to display the results of the algorithm as a list showing the selected links and the various data associated with the links. The links shown on the screen to the end user are active links, similar to the active comments used in the text boxes. The end user may instantaneously link to the destination node that the user selects. The list format provides link information in a style familiar to user of the Internet. However, this system is also capable of displaying the results in the user-friendly graphical format as described above. The graphical user interface program described previously uses box coloring and sizing to communicate large amounts of information quickly and intelligibly to the user. In a preferred embodiment, different colors for boxes are assigned depending on what type of node they represent (e.g., a web page, web site, a document, a file transfer protocol (FTP) (a common internet designation for news sites)). Preferably, the box is given depth. The amount of URL links a node contains may determine the amount of depth.

The graphical user interface program displays a list of the most related web pages to the source web page. This list includes documents, web sites, and pages which are directly or indirectly linked to the subject document or the subject topic. The links can be source links or influence links, so the end user may monitor the sites to which his site (the source web page) is referring, and the end user may view the sites which are referring to his site. The system can parse the URL of the destination nodes for a variety of information. Thus, the end user may monitor whether the connections to which his web site refers are still open, the end user may view the date and time a

destination node was modified, and the end user may view the identification of the organization or author of the destination node that directly or indirectly links to the source node. The GUI program displays all of this information either in the list format or in the text box used in the graphical format. Graphical comments may be placed in the text box to communicate information quickly, such as showing a happy face for a connected application, and so forth. Hyperlinks can appear as active comments in a text box in order to allow the user to instantaneously jump to the web page represented by the text box.

Although this computerized system for researching data is described as functioning in the World Wide Web environment, it can function equally well in any network system. A network that utilizes any type of hyperjump to connect documents together can serve as the links analyzed by this invention. This system therefore can be modified to navigate and search through internal company networks, and provide the same features as described above for the Web application. Additionally, the comment boxes can be tailored to display critical information about company files, thus enhancing its usefulness for the company employee who is attempting to sort through company documents stored on a network.

ADVANCED TELEVISION ENHANCEMENT FORUM CONTENT SPECIFICATION

Advanced Television (ATV) is the name given by the U.S. Federal Communications Commission to digital TV, the use of digital transmission of video and audio information on broadcast channels and cable TV. ATV includes both high-definition television (HDTV), a format for digital video compression, transmission, and presentation and also the creation of additional channels on the current analog 6 Mhz channel.

The Advanced Television Enhancement Forum (ATVEF) is a cross-industry group formed to specify a single public standard for delivering interactive television

experiences that can be authored once using a variety of tools and deployed to a variety of television, set-top, and PC-based receivers. This document specifies the content formats and delivery mechanisms that provide the kind of enhanced television experience that will meet the needs of the industry.

The Enhanced Content Specification is a foundation specification, defining fundamentals necessary to enable creation of HTML-enhanced television content so that it can be reliably broadcast across any network to any compliant receiver. The scope is narrowly defined as we strive to build agreement across the industries that are key to the success of enhanced television.

In general, the ATVEF specification for enhanced television programming uses existing Internet technologies. It delivers enhanced TV programming over both analog and digital video systems using terrestrial, cable, satellite and Internet networks. The specification can be used in both one-way broadcast and two way video systems, and is designed to be compatible with all international standards for both analog and digital video systems.

The ATVEF specification comprises three parts:

1. Content specifications to establish minimum requirements for receivers.
2. Delivery specifications for transport of enhanced TV content.
3. A set of specific bindings.

The ATVEF Specification was designed by a consortium of broadcast and cable networks, consumer electronics companies, television transport operators and technology companies to define a common, worldwide specification for enhanced television programming.

A central design point was to use existing standards wherever possible and to minimize the creation of new specifications. The content creators in the group determined that existing web standards, with only minimal extensions for television integration, provide a rich set of capabilities for building enhanced TV content in

today's marketplace. The ATVEF specification references full existing specifications for HTML, ECMAScript, DOM, CSS and media types as the basis of the content specification. Described below are the minimal requirements for content support for compliant receivers. The specification is not a limit on what content can be sent, but rather provides a common set of capabilities so that content developers can author content once and play on the maximum number of players.

Another key design goal was to provide a single solution that would work on a wide variety of networks. ATVEF is capable of running on both analog and digital video systems as well as networks with no video at all. The specification also supports transmission across terrestrial (over the air), cable, and satellite systems as well as over the Internet. In addition, it will also bridge between networks - for example data on an analog terrestrial broadcast must easily bridge to a digital cable system. This design goal was achieved through the definition of a transport-independent content format and the use of IP as the reference binding. Since IP bindings already exist for each of these video systems, ATVEF can take advantage of this work. Two transports are described below - one for broadcast data and one for data pulled through a return path.

While the ATVEF specification has the capability to run on any video network, a complete specification requires a specific binding to each video network standard in order to ensure true interoperability. Two bindings are also described below—the reference binding to IP and the example NTSC binding. The IP binding is the reference binding both because it provides a complete example of ATVEF protocols and because most networks support the IP protocol. The NTSC binding is included as an example of an ATVEF binding to a specific video standard. It is not the role of the ATVEF group to define bindings for all video standards. The appropriate standards body should define the bindings for each video standard - PAL, SECAM, DVB, ATSC and others.

There are many roles in the production and delivery of television enhancements. This document refers to three key roles: content creator, transport operator, and

receiver. The content creator originates the content components of the enhancement including graphics, layout, interaction and triggers. The transport operator runs a video delivery infrastructure (terrestrial, cable, satellite or other) that includes a transport for ATVEF data. The receiver is a hardware and software implementation (television, set-top box, or personal computer) that decodes and plays ATVEF content. A particular group or company may participate as one, two or all three of these roles.

Content Specifications

The ATVEF content specification provides content creators with a reliable definition of mandatory content support on all compliant receivers. In addition, any other kind of data content can be sent over ATVEF transport including HTML, VRML, Java, or even private data files. When a content creator wants to broadcast an enhancement to play on the maximum number of receivers, the data should conform to the content specification.

In the case where the content author knows the specific capabilities of the target receiver, data can be sent over ATVEF transport that is outside the content specification including DHTML, Java, or even private data files.

In the ATVEF specification, there is one defined content specification: level 1.0.

Content Level 1.0

Content Formats

The foundation for ATVEF content is existing web standards. Mandatory support is required for the following standard specifications:

HTML 4.0 (Frameset Document Type Definition)
CSS 1
ECMAScript
DOM 0

Note:

ECMAScript plus DOM 0 is equivalent to JavaScript 1.1.

Note: Receivers are required to supply 1KB for session cookies. Cookies support is not required to be persistent when a receiver is turned off.

Content Type Support

Because ATVEF supports one-way broadcast of data, content creators cannot customize the content for each receiver as they do today with two-way HTTP.

ATVEF specifies the following base profile of supported MIME types that must be supported in each receiving implementation:

text/html (HTML 4.0)

text/plain

text/css (CSS1 only)

image/png (no progressive encoding)

image/jpg (no progressive encoding)

audio/basic

Support for the following widely used MIME types is currently recommended in all receiving implementations for compatibility with existing content. Support is not required and content creators should take into account that the types may not be supported.

image/gif (no progressive encoding)

audio/wav

Integrating TV with Web Pages

Use the "tv:" URL to reference a broadcast television channel. The "tv:" URL may be used anywhere that a URL may reference an image.

Examples of "tv:" URL usage include the object, img, body, frameset, a, div and table tags.

The Trigger Receiver Object

TV enhancement HTML pages that expect to have triggers sent to them via an ATVEF trigger stream must use the HTML object tag to include a trigger receiver object on a page. The trigger receiver object, implemented by the receiver, processes triggers for the associated enhancement in the context of the page containing the object. The content type for this object is "application/tve-trigger". If a page consists of multiple frames, only one may contain a receiver object.

Sample instantiation:

<OBJECT TYPE="application/tve-trigger" ID="triggerReceiverObj">
</OBJECT>

Properties:

triggerReceiverObj.enabled	A Boolean, indicating if the triggers are enabled. The default value is true (read/write)
triggerReceiverObj.sourceId	A string containing the ASCII-hex encoded UUID for the announcement for this stream. sourceID is null if the UUID was not set for the enhancement. (read only)
triggerReceiverObj.releasable	A Boolean indicating that the currently displayed top level page associated with the active enhancement can be released and may be automatically replaced with a new resource when a valid trigger containing a new URL is received. Such a trigger must contain a [name:] attribute. The default value is false.
triggerReceiverObj.backChannel	A string indicating the availability and state of a backchannel to the Internet on the current receive When backChannel returns "permanent" or "connected," receivers can generally perform HT get or post methods and expect real-time response When backChannel returns "disconnected," recei can also expect to perform HTTP get or post met but there will be an indeterminate delay while a connection is established. When backChannel ret "unavailable," no HTTP get or post methods can

	performed. No standard behavior can be assumed when any other value is returned. Value is one of: permanent --Always connected connected --Currently connected, but not always disconnected --Not currently connected, but can connect unavailable --Never connected
triggerReceiverObj.contentLevel	A number that corresponds to the ATVEF content level of the receiver. For this specification, it is 1.0.

Triggers

Triggers are real-time events delivered for the enhanced TV program. Receiver implementations will set their own policy for allowing users to turn on or off enhanced TV content, and can use trigger arrival as a signal to notify users of enhanced content availability.

Triggers always include an URL, and may optionally also include a human-readable name, an expiration date, and a script. Receiver implementers are free to decide how to turn on enhancements and how to enable the user to choose among enhancements. Triggers that include a "name" attribute may be used to initiate an enhancement either automatically, or with user confirmation. The initial top-level page for that enhancement is indicated by the URL in that trigger. Triggers that do not include a "name" attribute are not intended to initiate an enhancement, but should only be processed as events which affect (through the "script" attribute) enhancements that are currently active. If the URL matches the current top-level page, and the expiration has not been reached, the script is executed on that page through the trigger receiver object. When testing for a match, parameters and fragment

identifiers (i.e. characters in the URL including and following the first "?" or "#" character) in an URL are ignored.

Triggers are text based, and their syntax follows the basic format of the EIZ-746 standard (7-bit ASCII, the high-order bit of the first byte must be "0"). Note: The triggers follow the syntax of EIA-746A, but may be transported in multicast IP packets or other transport rather than using the EIA-608 system.

All triggers defined in this version of ATVEF are text-based and must begin with ASCII '<'. All other values for the first byte are reserved. These reserved values may be used in the future to signal additional non-text based messages. Receivers should ignore any trigger that does not begin with the '<' in the first byte.

The general format for triggers (consistent with EIA-746A) is a required URL followed by zero or more attribute/value pairs and an optional checksum:

<url> **[***attr*$_1$**:** *val*$_1$**][***attr*$_2$**:***val*$_2$**]**...**[***attr*$_n$**:***val*$_n$**][***checksum***]**

Character set: All characters are based on ISO-8859-1 character set (also known as Latin-1 and compatible with US-ASCII) in the range 0x20 and 0x7e. Any need for characters outside of this range (or excluded by attribute limits below) must be encoded using the standard Internet URL mechanism of the percent character ("%") followed by the two-digit hexadecimal value of the character in ISO-8859-1.

The trigger begins with a required URL:

<url> The URL is enclosed in angle brackets (e.g. <http://xyz.com/fun.html>). Although any URL can be sent in this syntax, ATVEF content level 1 only requires support for http: and lid: URL schemes.

The following attribute/value pairs are defined:

[name:*string***]** The **name** attribute provides a readable text description (e.g.[name:Find Out More]). The *string* is any string of characters between 0x20 and 0x7e except

square brackets (0x5b and 0x5d) and angle brackets (0x3c and 0x3e). The name attribute can be abbreviated as the single letter "n" (e.g.[n:Find Out More]).

[expires:*time***]** The **expires** attribute provides an expiration date, after which the link is no longer valid (e.g.[expires:19971223]). The *time* conforms to the ISO-8601 standard, except that it is assumed to be UTC unless the time zone is specified. A recommended usage is the form *yyyymmdd*T*hhmmss*, where the capital letter "T" separates the date from the time. It is possible to shorten the *time* string by reducing the resolution. For example *yyyymmdd*T*hhmm* (no seconds specified) is valid, as is simply *yyyymmdd* (no time specified at all). When no time is specified, expiration is at the beginning of the specified day. The expires attribute can be abbreviated as the single letter "e" (e.g.[e:19971223]).

[script:*string***]** The **script** attribute provides a script fragment to execute within the context of the page containing the trigger receiver object (e.g.[script:shownews()]). The *string* is an ECMAScript fragment. The script attribute can be abbreviated as the single letter "s" (e.g.[s:shownews()]).An example of a script attribute used to navigate a frame within a page to a new URL: [script:frame1.src="http://atv.com/f1"]

The optional checksum must come at the end of the trigger. (Note: EIA-746A requires the inclusion of a checksum to ensure data integrity over line 21 bindings. In other bindings, such as IP, this may not be necessary, and is not required.)

[*checksum*] The checksum is provided to detect data corruption. To compute the checksum, adjacent characters in the string (starting with the left angle bracket) are paired to form 16-bit integers; if there are an odd number of characters, the final character is paired with a byte of zeros. The checksum is computed so that the one's complement of all of these 16-bit integers plus the checksum equals the 16-bit integer with all 1 bits (0 in one's complement arithmetic). This checksum is identical to that used in the Internet Protocol (described in RFC 791); further details on the computation of this checksum are given in IETF RFC 1071. This 16-bit checksum is transmitted as four hexadecimal digits in square brackets following the right square bracket of the final attribute/value pair (or following the right angle bracket if there are no attribute/value pairs). The checksum is sent in network byte order, with the most significant byte sent first. Because the checksum characters themselves (including the surrounding square brackets) are not included in the calculation of the checksum, they must be stripped from the string by the receiver before the checksum is recalculated there. Characters outside the range 0x20 to 0x7e (including the second byte of two-byte control codes) shall not be included in the checksum calculation.

Other attributes could be defined at a later date. However, all other single character attribute names are reserved. Receivers should ignore attributes they do not understand.

Using the description above, all the following are valid trigger strings:

<http://xyz.com/fun.html>

<http://xyz.com/fun.html>[name:Find out More!]

<lid://xyz.com/fun.html>[n:Find out More!]

<lid://xyz.com/fun.html>[n:Fun!][e:19991231T115959] [s:frame1.src="http://atv.com/frame1"]
<http://WWW.newmfr.com>[name:New][C015]

Note: If a trigger does not contain a [name:] attribute, the enhancement referenced by the trigger should not be presented to the user.

The Local Identifier URL Scheme ("lid:")

Content delivered by a one-way broadcast is not necessarily available on-demand, as it is when delivered by HTTP or FTP. For such content, it is necessary to have a local name for each resource. To support cross-references within the content (for use in hyperlinks or to embed one piece of content in another), these local names must be location-independent.

The "lid:" URL scheme enables content creators to assign unique identifiers to each resource relative to a given namespace. Thus the author can establish a new namespace for a set of content and then use simple, human-readable names for all resources within that space. The "lid:" scheme is used by the "Content-Location:" field in the UHTTP resource transfer header to identify resources that should be stored locally by a broadcast capable receiver platform and are not accessible via the Internet.

The syntax of the "lid:" URL is as follows:

lid://{namespace-id}[/{resource-path}]

The {namespace-id} specifies a unique identifier (e.g. UUID or a domain name) to use as the namespace for this content or as a root for the URL. The {resource-path} names a specific resource within the namespace, and must follow the generic relative URL syntax. As with all URL schemes that support the generic relative URL syntax, this path component can be used alone as a relative URL, where the namespace is implied by a base URL specified for the content through other means.

While all compliant implementations of enhanced TV receivers support absolute URLs within the UHTTP header and broadcast triggers, some implementations may not correctly process absolute URLs using the "lid:" scheme within HTML content.

To ensure that HTML content is correctly interpreted by these receiving platforms, content should use only relative URLs in their HTML. Triggers use the full "lid:" URL to load the top level HTML page and that page uses relative URLs to refer to other resources.

Some examples:

lid://unique2345@blahblah.com/rootimage.jpg

lid://xyz.com/myshow/episode100/george.html

lid://12abc554c3d3dd3f12abc554c3d3dd3f/logos/ourlogo.gif

The first example uses a RFC822ftp://ftp.isi.edu/in-notes/rfc822.txt message-id style unique id, the second one uses a domain name as a unique identifier, and the third uses a text encoding of an UUID. Each is a valid mechanism for describing a "lid:" namespace.

Content Caching

Receivers must be able to support one megabyte (1 MB) of cached simultaneous content. Content creators who want to reach the maximum number of receivers should manage their content to require a high-water mark of simultaneous cached content of 1 MB or less. The specific cache size required for each enhancement must be specified in the announcement.

tve-size represents the maximum size cache needed to hold content for the current page at any time during the program and also all pages reachable by local links. It is the high water mark during the program, not the total content delivered during the program. Size is measured as the size when the content is delivered (after decompression for content sent using gzip or other compression techniques).

Additional Content Levels

In the ATVEF spec, there is only one defined content specification--level 1.0. The content level of the client is available via ECMAScript using the receiverObj.contentLevelhttp://www.atvef.com/library/ - contentlevel property, and can be used in announcements. Possible directions for future content levels include

Dynamic HTML, synchronized multimedia, 3-D rendering, tuning, XML, Java, and higher-quality audio among others.

Transport Specifications

The display of enhanced TV content consists of two steps: delivery of data resources (e.g. HTML pages) and display of named resources synchronized by triggers. All forms of ATVEF transport involve data delivery and triggers. The capability of networks for one-way and/or two-way communication drives the definition of two models of transport.

ATVEF defines two kinds of transport. Transport A is for delivery of triggers by the forward path and the pulling of data by a (required) return path. Transport B is for delivery of triggers and data by the forward path where the return path is optional.

Transport Type A: Return-path Data

Most broadcast media define a way for data service text to be delivered with the video signal. In some systems, this is called closed captioning or text mode service; in other systems, this is called teletext or subtitling. For the sake of this discussion, triggers delivered over such mechanisms will be generically referred to as broadcast data triggers.

Some existing broadcast data services provide a mechanism for trigger delivery, but not resource deliver, due to limited bandwidth. Content creators may encode broadcast data triggers using these. Broadcast data streams only contain broadcast data triggers so there is no announcement or broadcast content delivery mechanism. Because there are no announcements, the broadcast data service stream is considered to be implicitly announced as a permanent session.

In addition to the other attributes used in triggers, ATVEF transport type A triggers must contain an additional attribute, "tve:". The "tve:" attribute indicates to the receiver that the content described in the trigger is conformant to the ATVEF content specification level. For example, [tve:1.0]. The "tve:" attribute can be abbreviated as

the single letter "v". The version number can be abbreviated to a single digit when the version ends in ".0" (e.g.[v:1] is the same as [tve:1.0]). The "tve:" attribute is equivalent to the use of "type:tve" and "tve-level:" in SAP/SDP announcements in the transport type B IP multicast binding. This attribute is ignored if present in a trigger in transport B since these values are set in transport type B in the announcement. If the "tve:" attribute is not present in a transport type A trigger, the content described in the trigger is not considered to be ATVEF content.

Television transport operators should use the standard mechanisms for broadcast data trigger transmission for the appropriate medium (EIA, ATSC, DVB, etc.). It is assumed that when the user tunes to a TV channel, the receiver locates and delivers broadcast data triggers associated with the TV broadcast. Tuning and decoding broadcast data triggers is implementation and delivery standard specific and is specified in the appropriate ATVEF binding. A mechanism must be defined for encoding broadcast data triggers for each delivery standard. For example in the NTSC binding, the broadcast data trigger syntax is encoded on the Text2 (T2) channel of line 21 using the EIA-746A system.

Because there is no content delivery system, broadcast data triggers usually require two-way Internet connections to fetch content over HTTP.

Note: Television transport operators and content creators need to plan to handle the scalability issues associated with large numbers of HTTP requests responding at roughly the same time to broadcast triggers.

Transport Type B: Broadcast Data

Transport type B is for true broadcast of both the resource data and triggers. As such, transport type B can run on TV broadcast networks without Internet connections, unlike transport type A. An additional Internet connection allowing a return path can be added to provide two way capabilities like e-commerce or general Web browsing.

Transport type B uses announcements to offer one or more enhancements of a TV channel. An announcement specifies the location of both the resource stream (the files that provide content) and the trigger stream for an enhancement. Multiple enhancements can be offered as choices that differ on characteristics like language or required cache size or bandwidth. In addition to locating the files and trigger streams, announcements must be able to provide the following information: language, start and stop times, bandwidth, peak storage size needed for incoming resources, ATVEF content level the resources represent, an optional UUID that identifies the content, an optional string that identifies the broadcast channel for systems that send ATVEF content separately from the audio/video TV broadcast. The receiver must be able to start receiving data from only the description broadcast in the announcement.

Transport type B also requires a protocol that provides for delivery of resources. In one way broadcast systems, this is a one way resource transfer protocol that allows for broadcast delivery of resources. The resource delivered, no matter what the resource transfer method, must include HTTP headers to package the file on the resource transfer protocol. All resources delivered using resource transfer are named using URLs. These resources are then stored locally, and retrieved from this local storage when referenced using this same URL. All receivers must support local storage and retrieval of content using the "lid:" URL scheme and the familiar "http:" URL scheme. When "lid:" is used, the resources are delivered only through broadcast and are not available on demand. When "http:" is used, the resources that are delivered through broadcast also exist on the World Wide Web and can be requested from the appropriate server using standard HTTP. Sending "http:" resources using resource transfer effectively pre-loads the local cache, thus avoiding large numbers of simultaneous hits on Web servers when those same resources are requested by many receivers. Furthermore, this mechanism allows receivers to view the same content that appears on the Web even when no Internet connection is available. Content creators can freely mix resources that use either the "lid:" or "http:" schemes in the same enhanced broadcast. Because the underlying resource transfer protocol is not limited to carrying resources named by any particular URL scheme, some

receivers will store and retrieve content named using other URL schemes, such as "ftp:", as well as the required "lid:" and "http:".

Transport type B uses the same syntax for triggers as type A.

The "ATVEF Reference Binding for IP Multicast" describes three protocols based on IP multicast transmission for each of the three data streams: 1) announcements; 2) triggers; and 3) one-way resource transfer.

Simultaneous Support of Transports A and B

A single video program may contain both transport type B (e.g. IP) and transport type A (e.g. broadcast data triggers) simultaneously. This is advantageous in order to target both IP-based receivers as well as receivers that can only receive broadcast data triggers.

Receivers may choose to support only IP based trigger streams and ignore broadcast data triggers, or receivers may support broadcast data triggers in the absence of IP based triggers, or receivers may support broadcast data triggers and IP based triggers simultaneously. For receivers that provide simultaneous support, ATVEF specifies the following behavior, which is identical to the treatment of IP based triggers on an active stream.

When a broadcast data trigger is encountered, its URL is compared to the URL of the current page. If the URLs match and the trigger contains a script, the script should be executed. If the URLs match but there is no script, the trigger is considered a re-transmission of the current page and should be ignored. If the URLs do not match and the trigger contains a name, the trigger is considered a new enhancement and may be offered to the viewer. If the URLs do not match and there is no name, the trigger should be ignored.

ATVEF Bindings

An ATVEF binding is a definition of how ATVEF runs on a given network. The binding may support either or both Transport types A and B. Having one standard ATVEF binding for each network is necessary so that receivers and broadcast tools can be developed independently.

The measure of a sufficient ATVEF binding is that all the data needed to build a compliant, interoperable receiver for a given network should be contained in the ATVEF spec, the network spec and the ATVEF network binding, if needed. Put another way, the ATVEF binding provides the glue between the network spec and the ATVEF spec, in cases where the network specification doesn't contain all the necessary information.

ATVEF defines the Binding to IP as the *reference binding*. This is because IP is available to run over virtually any kind of network in existence. That means that one approach to building an ATVEF binding for a particular network is to simply define how IP is run on that network associated with a particular video program. The IP Binding can also be used as a model for a complete, compliant and efficient ATVEF binding.

This portion also includes an example of a binding to a specific network standard--the ATVEF Binding to NTSC. This binding can be used as a model for how to build an ATVEF binding to a specific video standard. The example NTSC binding defines transport type A using an NTSC-specific method and defines transport type B using the IP reference binding. It is not the role of the ATVEF group to define bindings for all video standards. The appropriate standards body should define the bindings for each video standard--PAL, SECAM, DVB, ATSC and others.

ATVEF Binding to IP Multicast (Reference Binding)

IP multicast is the mechanism for broadcast data delivery. Content creators should assume IP addresses may be changed downstream, and therefore should not use them in their content. The transport operator is only responsible for making sure that an IP address is valid on the physical network where they broadcast it (not for any re-

broadcasting). When possible, content creators should use valid IP multicast addresses to minimize the chance of collisions. Some systems may have two-way Internet connections. Capabilities in those systems are outside the scope of this document and are described by the appropriate Internet standards.

Transport operators should use the standard IP transmission system for the appropriate medium (IETF, ATSC, DVB, etc.). It is assumed that when the user tunes to a TV channel, the receiver automatically locates and delivers IP datagrams associated with the TV broadcast. The mechanism for tuning video and connecting to the appropriate data stream is implementation and delivery standard specific and is not specified in this framework.

Announcement Protocol

Announcements are used to announce currently available programming to the receiver. The IP multicast addresses and ports for resource transfer and for triggers are announced using SDP announcements (RFC 2327). The SDP Header is preceded by an 8-byte SAP header. SAP is still in Internet Draft form, but the non-optional first 8 bytes are stable (see http://WWW.ietf.org/html.charters/mmusic-charter.html). Announcements are sent on a well-known address (224.0.1.113) and port (2670). This address and port have been registered with the IANA.

v=0	SDP Version, required to be 0.
o=username *sid version* IN IP4 *ipaddress*	Owner & session identifier, defined as usual in SDP spec. Username is "-", network type is IN, address type is IP4. SessionID identifies an announcement for a particular broadcast (it can be a permanent announcement for all programming on a broadcast channel or for a particular show). Version indicates the version of the message. These values allow receivers to match a message to a previous message and know whether it has changed. Session ID and

	Version should be NTP values as recommended in SDP.
s=name	Session name, required as in SDP spec.
i=, u=	Optional, as in SDP spec.
e=, p=	E-mail address or phone number, at least one required in SDP spec.
b=CT:*number*	Optional in SDP spec, but Required here. Bandwidth in kbps as in the SDP spec. Bandwidth of the broadcast data can be used by receivers to choose among multiple versions of enhancement data according to the bandwidth the receiver can handle.
t=*start stop*	As in SDP spec gives start and stop time in NTP format. With programs stored on tape, at times it will not be possible to insert new announcements, so start times on tape could be incorrect. In this case, the start time should be set to the original broadcast time and the stop time set to 0. This is the standard for an unbounded session. Assumptions are then made about the stop time (see RFC 2327). A new announcement for a new program for the same broadcast station replaces the previous one. It is preferred that a tool read the tape and generate announcements with correct start and stop times, but not required. Content creators can choose to use only a station ID and not provide information about individual programs.
a=UUID:*UUID*	Optional. The UUID should uniquely identify the

	enhancement (for example, a different UUID for each program), and can be accessed using the trigger receiver object. In analog TV and many types of digital TV broadcast data is tied tightly to A/V. Each virtual channel has its own private network associated with it. In other systems, enhancements for many virtual channels can be carried on the same network. These systems can use the UUID to link a TV broadcast with a particular enhancement. How that association is indicated is beyond the scope of this document. One technique would be to place the UUID in electronic program guide information. Use ASCII HEX to encode UUIDs.
a=type:tve	Required. Indicates to the receiver that the announcement refers to an ATVEF enhancement.
a=*lang*, a=*sdplang*	Optional, as in SDP spec.
a=tve-type:<types>	Optional. tve-type: specifies an extensible list of types that describe the nature of the enhancement. It is a session-level attribute and is not dependent on charset. a=tve-type:primary Optional. tve-type:primary specifies that this will be the primary enhancement stream associated with the currently playing video program whenever this enhancement's trigger stream is active. If tve-type:primary is not specified, the TVE stream is never the primary enhancement stream associated with video. This, like all tve-type: attributes, is a session level attribute. This attribute can be used by receivers to implement automatic loading of primary video enhancement

	streams. The actual display of and switching between enhancement streams is handled by the trigger streams.
a=tve-size:*Kbytes*	Required. tve-size: provides an estimate of the high-water mark of cache storage in kilobytes that will be required during the playing of the enhancement. This is necessary so that receivers can adequately judge whether or not they can successfully play an enhancement from beginning to end.
a=tve-level:*x*	Content level identifier, where x is 1.0 for this version of the framework (optional, default is 1.0).
a=tve-ends:*seconds*	Optional, specifies an end time relative to the reception time of the SDP announcement. *Seconds* is the number of seconds in the future that this announcement is valid. *Seconds* may change (count down) as an announced session progresses. This attribute, when present, overrides the default assumptions for end times in unbounded announcements.
m=data *portvalue*/2 tve-file/tve-trigger c=IN IP4 *ipaddress/ttl*	As in SDP spec. Compact form specifying 2 ports on same address

When there are multiple alternative enhancement streams for the same video program, they must all be announced at the media level of the same SDP announcement. All enhancement streams announced in the same SDP announcement are considered to be mutually exclusive variants of the primary enhancement stream. The receiver can choose between them based on media level attributes. For example,

the a=lang field can be used at the media level to choose between language variants of the primary enhancement.

Each media section for the tve-file media type begins the next enhancement definition.

A longer form is available if the content creator or transport operator wants to use different IP addresses and ports for the data stream and trigger stream:

m=data *portvalue* tve-file c=IN IP4 *ipaddress/ttl*	Alternative form for specifying addresses and ports (for file protocol, as in SDP spec)
m=data *portvalue* tve-trigger c=IN IP4 *ipaddress/ttl*	For control protocol, as in SDP spec.

Announcement Example:

```
v=0
o=-2890844526 2890842807 IN IP4 tve.niceBroadcaster.com
s=Day & Night & Day Again
i=A very long TV Soap Opera
e=help@niceBroadcaster.com
a=UUID:f81d4fae-7dec-11d0-a765-00a0c91e6bf6
a=type:tve
a=tve-level:1.0
t=2873397496 0
a=tve-ends:30000
a=tve-type:primary
m=data 52127/2 tve-file/tve-trigger
c=IN IP4 224.0.1.112/127
b=CT:100
a=tve-size:1024
m=data 52127/2 tve-file/tve-trigger
c=IN IP4 224.0.0.1/127
```

b=CT:1024
a=tve-size:4096

Trigger Protocol

The trigger protocol carries a single trigger in a single UDP/IP multicast packet. Triggers are real-time events broadcast inside IP multicast packets delivered on the address and port defined in the SDP announcement for the enhanced TV program. The trigger protocol is thus very lightweight in order to provide quick synchronization.

Resource Transfer: UHTTP

A one-way IP multicast based resource transfer protocol, the Unidirectional Hypertext Transfer Protocol (UHTTP) is defined. UHTTP is a simple, robust, one-way resource transfer protocol that is designed to efficiently deliver resource data in a one-way broadcast-only environment. This resource transfer protocol is appropriate for IP multicast over television vertical blanking interval (IPVBI), in IP multicast carried in MPEG-2, or in other unidirectional transport systems.

Web pages and their related resources (such as images and scripts) are broadcast over UDP/IP multicast along with their related TV signal. An announcement broadcast by the TV station tells the receiver which IP multicast address and port to listen to for the data. The only data broadcast to this address and port are resources intended for display as Web content.

While HTTP headers preceding resource content are optional in the UHTTP protocol, they are required when the protocol is used for ATVEF enhanced TV. Compliant receivers must support content encodings of "gzip" as specified by the "Content-Encoding" HTTP header field.

ATVEF Binding to NTSC

In NTSC, ATVEF data is broadcast by encoding bytes in the vertical blanking interval of individual video fields. Two different techniques are used for broadcasting data using ATVEF transport A and ATVEF transport B.

Transport A: VBI Line 21

ATVEF triggers are transmitted on VBI Line 21 of the NTSC signal using the T-2 service as specified in EIA-608. This encoding is consistent with the EIA-746A specification which describes how to send URLs and related information on VBI line 21 of an NTSC channel, without interfering with other data (e.g., closed captions) also sent on that line. The checksum described in the ATVEF trigger definition is required in the Transport A ATVEF Binding to NTSC.

Note that, as specified in the ATVEF trigger definition, triggers are encoded using ISO-8859-1 and not the EIA-608 character set. (While most characters are the same in both encodings, a few codes have different meanings.)

ATVEF trigger length should be kept as short as possible. ATVEF trigger transmissions should be limited to 25% of the total field 1 bandwidth, even if more bandwidth is available after captioning, to allow for other downstream services.

Transport B: IP over VBI

IP datagrams should be sent according to the specification drafted by the IP over VBI working group of the Internet Engineering Task Force (see http://WWW.ietf.org/html.charters/ipvbi-charter.html). Note that this specification is currently in late draft stage, but is expected to be completed and published as a standards-track document in the coming weeks. In NTSC, the NABTS (rather than WST) byte encoding should be used.

ATVEF IP streams should be sent on the packet addresses 0x4b0 through 0x4bf. Other packet addresses may be used, but receivers are only required to handle IP datagrams arriving using packet addresses 0x4b0 through 0x4bf.

Unidirectional Hypertext Transfer Protocol (UHTTP)

The Unidirectional Hypertext Transfer Protocol, or UHTTP, is a simple, robust, one-way data transfer protocol that is designed to efficiently deliver resource data in a one-way broadcast-only environment. This transfer protocol is appropriate for one-way IP multicast over television vertical blanking interval (IP/VBI) or other unidirectional transport systems.

This portion describes the format of the message packets that carry UHTTP data. It describes the information needed to create the messages using the protocol on the broadcast side and to turn those messages back into resources on the receiving side.

Resources sent using the UHTTP protocol are divided into a set of packets, encapsulated in UDP. Typically, these packets may be delivered via multicast IP, but this is not required. Each packet contains enough header information to begin capturing the data at any time during the broadcast, even midway through the transfer. This header contains an identifier (in the form of an UUID) that uniquely identifies the transfer, and additional information that enables the receiver to place the data following the header in the appropriate location within the transfer. Additional information indicates to the receiver how long to continue listening for additional data.

UHTTP includes the ability to gather segments over multiple retransmissions to correct for missing packets. It is also possible to group resources together for all-or-none delivery within a UHTTP transfer. The protocol also includes a forward error correcting mechanism which provides for the ability to restore missing data in the event of limited packet loss.

Data Transfer Features Enabled by the UHTTP Protocol

Robust Delivery: Gathering data over multiple transmissions

Data can be resent via UHTTP using the same globally unique TransferID. The data is delivered as individual segments, each of which is in a UDP message, potentially delivered via IP multicast. Information in the header allows a receiving application

to receive segments out of order or multiple times. If the transfer data is sent repeatedly, the receiving service can fill in missing ranges using these retransmissions. This provides robust (though not necessarily reliable) data delivery. Additionally, forward-error correction (FEC), using an XOR algorithm, provides for recovery of some missing segments in the face of segment loss without re-transmission.

Meta-information in the form of HTTP-style headers

The protocol provides for the inclusion of HTTP-style headers preceding the resource data. These headers may include information describing the content type of the resource and content location in the form of a URL. It may also be used to describe groups of resources as a multipart construction. Other meta-information, including date stamping and expiration dates, may be used to provide additional information about the resource content.

UHTTP Header Details

The UHTTP header is at the start of every UHTTP IP/UDP multicast payload. All values are network byte order. The fields are as follows:

Name	Size	Description
Version	5 bits	Describes the version of the protocol. The protocol described here is version 0.
ExtensionHeader	1 bit	When set, this bit indicates that one or more extension header fields are present.
HTTPHeadersPrecede	1 bit	A bit flag that, when set to 1, indicates that HTTP-style headers precede the resource data. These HTTP-style headers are considered part of the data when calculating the ResourceSize and SegStartByte fields, as well as for forward error correction. This bit must be set in all

		packets associated with a UHTTP transfer when HTTP-style headers precede the data. When set to zero, no HTTP-style headers precede the resource data.
CRCFollows	1 bit	When the CRCFollows bit is set to 1, a 32 bit CRC is calculated and can be used to detect possible corruption in the data delivered via UHTTP. Using the MPEG-2 CRC algorithm, the CRC is calculated on the complete data, including HTTP-style headers, if any. It is then appended to the end of the data in the last logical packet. This CRC field is considered part of the data for the purposes of calculating the resource length and calculating the forward error correction. The bit must be set in all packets associated with a UHTTP transfer when a CRC is used.
PacketsInXORBlock	1 byte	Describes the number of packets in a forward error correction block, including the forward error correction packet. Set to zero when no forward error correction is used.
RetransmitExpiration	2 bytes	Time in seconds over which the resource may be retransmitted. This indicates how long the receiving software should wait to try to recover missing packets that follow in retransmissions of the same resource. This allows a resource to be carouseled, or sent repeatedly to increase the chances of

		delivery without missing segments. Set to zero if the resource will not be retransmitted. Set to maximum if the software should continue listening. The RetransmissionExpiration field should be updated to remain accurate during retransmissions, including the current transmission.
TransferID	16 bytes	Globally unique identifier (UUID) for the UHTTP transfer. This ID allows receiving software to identify which segments correspond to a given transfer, and determine when re-transmission occurs.
ResourceSize	4 bytes	Size of the complete resource data itself (excluding segment headers, XOR segments and padding for exclusive-or correction). This length does include the length of the HTTP-style headers, if any, as well as the 4-byte CRC, if the CRCFollows bit is set to 1.
SegStartByte	4 bytes	Start byte in the transfer for this data segment. When XOR data is used to replace missing packets, SegStartByte includes the XOR data as well as the resource data, and optional HTTP-style headers and CRC. This allows for determining where all packets fit regardless of delivery order. The exclusive-or correction packet looks like any other UHTTP packet. Its data payload is simply

		the exclusive-or of a number of packets that precede it in order in the data. The number of packets in an XOR block is specified in the PacketsInXORBlock field described above.
Extension Headers		Extension headers, if any.
Data Payload		The data payload for the UHTTP transfer, including HTTP-style headers, if any, and body.
Total Length:	28 bytes	

The UDP packet data length for the enclosing UDP packet is used to determine the length of the segment. It is permissible to send a packet that contains UHTTP header (and optional extension headers), but without any data. If no data is included, then the SegStartByte field is ignored.

UHTTP Extension Headers

If the ExtensionHeader flag is set in a UHTTP packet, additional optional header fields are present. These fields appear directly after main UHTTP header. Extension headers are optional on a packet-by-packet basis, and may appear on none, some or all of the UHTTP packets transmitted, depending on the ExtensionHeaderType. This specification defines a single extension header type, HTTPHeaderMap. Any extension headers with an unknown type should be ignored by receivers. The format for the fields within a UHTTP extension header are as follows:

Name	Size	Description
ExtensionHeaderFollows	1 bit	When 1, this field indicates that another extension header follows this

		one. When 0, the UHTTP data payload follows this extension header.
ExtensionHeaderType	15 bits	Identifies the extension header type. (1 = HTTPHeaderMap, all other values reserved).
ExtensionHeaderDataSize	2 bytes	Describes the length of the complete Extension Header data in bytes. Zero indicates that there is no ExtensionHeaderData following.
ExtensionHeaderData		The variable length data for this extension header. The length of the ExtensionHeaderData field is indicated by the ExtensionHeaderDataSize.

If the ExtensionHeaderFollows bit is set, then another ExtensionHeader follows this header. If the bit is cleared, then the UHTTP data payload follows the ExtensionHeaderData (if any) immediately.

HTTPHeaderMap Extension Header

One ExtensionHeaderType is defined for this specification. When ExtensionHeaderType is set to a value of 1, then the ExtensionHeaderData field contains an HTTPHeaderMap. A HTTPHeaderMap extension header may optionally be included whenever the UHTTP transfer contains HTTP-style header information (as indicated by the HTTPHeadersPrecede bit in the main UHTTP header). If HTTPHeaderMap extension headers are used, they should be included in every packet in a UHTTP transfer that contains header, body or forward-error correction (FEC) data.

The HTTPHeaderMap consists one or more sets of the following fields:

Name	Size	Description
HTTPHeaderStart	4 bytes	This field indicates an offset into the UHTTP data, in bytes, where a HTTP-style header is found. The offset is calculated from the beginning of the corrected UHTTP data, and does not include the FEC data when the FEC mechanism is used.
HTTPHeaderSize	4 bytes	This field indicates the length of the HTTP-style header, in bytes, including the HTTP-style header fields, the terminating pair of newline characters, and any preceding multipart boundary lines.
HTTPBodySize	4 bytes	This field indicates the length of the data body, in bytes, associated with the HTTP header described in this map entry.

When the UHTTP transfer consists of a single (i.e. non-multipart) resource, a single 12 bytes set of HTTPHeaderMap fields is present in the HTTPHeaderMap. The HTTPHeaderStart, in this case, will be set to zero and the HTTPHeaderSize will be set to the sum of the length of the HTTP-style header fields and all separating newline characters. The HTTPBodySize field will contain the size, in bytes, of the body data related to that header field.

When a UHTTP transfer contains multiple resources (as specified by a multipart content-type), multiple sets of HTTPHeaderMap groups may be included in the HTTPHeaderMap data, each indicating the offset and size of the HTTP-style headers for each resource, (including any multipart boundary lines, HTTP-style header fields and separating newline characters), as well as the size of the body relating to each header.

When including HTTPHeaderMap data, senders must at a minimum include HTTPHeaderMap entries for each HTTP-style header that is partially or completely included in a given packet. Additionally, when forward-error correction is used in UHTTP transfers that contain HTTPHeaderMaps extension headers, senders must include HTTPHeaderMap entries as extension headers in FEC-data packets for all HTTP-style header sections that may be corrected by the FEC packet. Senders are free to include additional HTTPHeaderMap entries in any packet beyond the minimum.

Forward Error Correction Mechanism

In addition to the ability to retransmit data and have missing segments filled in during retransmissions, this transfer protocol can also include extra data packets that can be used for simple missing packet error correction. When PacketsInXORBlock is set to zero, there is no exclusive-or forward error correction. When non-zero, all segments must be the same length. It is permissible to send packets with no data payload (but with UHTTP headers and optional extension headers). In this case, the packet is ignored in the calculation of forward error correction.

In blocks of PacketsInXORBlock equal size data segments, the last data segment in the block contains the exclusive-or of the preceding segments (PacketsInXORBlock -1). Each byte of the data in this "XOR segment" is the exclusive-or of the corresponding byte in each of the other segments in that data block. If the data is thought of as laid out separated into consecutive segments, then after every PacketsInXORBlock -1 segments another segment is inserted that looks exactly like resource data and has its own position offset into the transfer like resource data. The data in that segment is the exclusive-or of the previous packets in that block. If this technique is used, the data payload of all packets must be the same size. The packet containing the end of file data (including the optional CRC) must be zero filled. Packets between this packet and the last XOR packet need not be sent since the receiver knows their contents are all zeros since it knows the overall length. If they are sent they must be zero filled after the segment header. The last XOR packet has

the value SegStartByte calculated to be just as if zero filled extra packets were sent, but there is no requirement to send those empty packets.

To correct for a single missing packet in a block, the receiver should calculate the exclusive-or the data payload of the packets that arrived with the XOR data segment for that block. A key point is that segments can be sent in any order since each segment including the XOR segment indicate where in order they belong. By sending segments (including the XOR packets) out of order, there is protection against burst errors that lose successive packets. When re-transmitting a UHTTP transfer, a different order of segments can be used in each retransmission to avoid different types of burst errors. This protocol allows the headend (broadcast side) tools to decide how to order sending packets providing a great deal of flexibility. The receiving side does not need to know the transmission order (consistent with the fact that in general it cannot know the transmission order for IP multicast delivery). XOR data in the XOR packet is the exclusive-or of data segment contents only, including the HTTP-style header fields but not including the UHTTP header that is also in the packet.

HTTP-style headers used in UHTTP

The UHTTP transfer protocol can be used to deliver resources via a broadcast medium, which can simultaneously deliver resources, including web-related content, to large numbers of users simultaneously. HTTP-style headers are optional in UHTTP, but are required for resources intended to be interpreted as web content.

HTTP-style headers (HTTP 1.1) are required to precede the resource contents just as HTTP does when resources are sent as a response to a HTTP GET or POST command. The HTTP-style headers may provide additional information to the browser like the expiration time for the resource. The HTTP-style headers precede the body of the resource data, and are treated as part of the content. The protocol header and its version imply the equivalent HTTP response line (e.g. "HTTP/1.1 200 OK"). The header fields that are required to be supported by all receiving clients are

listed below and should be interpreted per the HTTP 1.1 specification. No assumption should be made for support of other header fields.

Supported HTTP-style headers

HTTP-style header Fields required for every resource:

Content-Length:

Content-Location:

Recommended HTTP-style header fields:

Content-Type:

Optional HTTP-style header fields:

Content-Base:

Content-Encoding:

Content-Language:

Content-Style-Type:

Date:

Expires:

Last-Modified:

Receivers will decode the headers and data and store them in a local cache system. Different platforms will have different cache sizes for storing local resources, which may or may not correspond to traditional browser caches. The use of "Content-Location:" headers with "lid:" style URLs is intended to mirror resource delivery to a local cache without requiring that the data be available on the web.

Receiving platforms should take into consideration that the same resources will likely be sent repeatedly to provide resources for users who tune in late. HTTP-style header fields can be examined to determine if the resource is already present, and so can be ignored. The "Date:", "Expires:", and "Last-Modified:" headers can be used to determine the lifetime of a resource in a given browser's cache.

When the "http:" scheme is specified in the URL, the HTTP-style header will contain the same information as the get response plus the "Content-Location:".

Packaging more than one resource

The HTTP "Content-Type:" field can be multipart/related. When this type is used, the HTTP-style header is ended as usual and is followed by the usual boundary structure for "multipart/related" separating multiple related resources that each use the HTTP-style header formats. This is a mechanism to package multiple related resources together in a single all-or-nothing transfer. The HTTP-style headers for individual subparts describe only the subpart, but are interpreted as per the HTTP 1.1 specification. In this case, it may be convenient to specify a "Content-Base:" for the entire package and then specify relative URLs for each of the "Content-Location:" headers for subsequent subparts.

The "multipart/related" content type should be used as per the IETF RFC 2387, with the following exceptions. The "start" and "start-info" attributes of the content-type header, which is optional in RFC 2387, are not supported.

An example using HTTP scheme URLs:

Content-Base: http://WWW.blahblah.com/
Content-Length: 3495
Content-Type: Multipart/Related; boundary=example98203804805
—example98203804805
Content-Location: http://WWW.blahblah.com/resource1.html
Content-Length: 495
Content-Type: text/html
Resource data for resource1.html
...<IMG src="image.jpg">...
—example98203804805
Content-Location: /image1.jpg
Content-Length: 1495
Content-Type: image/jpeg
Resource data for image1.jpg
—example98203804805

An identical example using "lid:" style URLs and relative URLs:

Content-Base: lid://unique2345@blahblah.com/

Content-Length: 3495

Content-Type: Multipart/Related; boundary=example98203804805

--example98203804805

Content-Location: resource1.html

Content-Length: 495

Content-Type: text/html

Resource data for resource1.html

...<IMG src="image.jpg">...

--example98203804805

Content-Location: image.jpg

Content-Length: 1495

Content-Type: image/jpeg

Resource data for image1.jpg

--example98203804805

While various embodiments and applications of this invention have been shown and described, it will be apparent to those skilled in the art that various modifications are possible without departing from the inventive concepts described herein. The invention, therefore, is not to be restricted except in the spirit of the appended claims.